ILMENAU UNIVERSITY OF TECHNOLOGY

Ilmenau University of Technology

Computer Science and Automation Faculty
Institute for Automation and Systems Engineering
Simulation and Optimal Processes

MASTER THESIS

"Comparison of hierarchical and distributed optimization for model predictive control"

to obtain the academic degree

Master of Science

Course of studies

"Electrical Power and Control Engineering"

submitted by

B.Sc. Evgeny Lazutkin

Student number: 47698

Responsible university lecturer: Prof. Dr.-Ing. habil. Pu Li
Supervisor: Dr.-Ing. Siegbert Hopfgarten

Ilmenau, Germany
1 October, 2012

Acknowledgements

I would like to thank everyone who has been supported me in creating this work.

I am especially grateful to my supervisors Prof. Dr.-Ing. habil. Pu Li and Dr.-Ing. Siegbert Hopfgarten, and ex-supervisor Dipl.-Ing. Martin Bartl for the opportunity to write this thesis at the Institute for Automation and Systems Engineering, Simulation and Optimal Processes Group. Thank you for your valuable time which I took in advantage. Your experience and support was an important asset.

Many big thanks to my father Yuri, mother Tatiana, brother Dmitry, grandparents **Vitaly** and **Maria**, and my girlfriend Angelika who have given me a lot of support during this time.

I also would like to thank my friends who live in Russia, especially Pavel, Dmitry, Natalia and Vladimir, Olesya and Denis for your support and that you stayed with me even when we live apart from each other.

Contents

1. Introduction

One of the modern approaches to the analysis and synthesis of control systems based on mathematical optimization techniques is the control theory of dynamic objects using predictive models - *Model Predictive Control* (MPC). Development of this approach [1] has begun in the early 60s for the control of processes and equipment in the petrochemical and energy production, for which the use of traditional methods of synthesis was extremely difficult due to the extraordinary complexity of their mathematical models. The study and application of MPC is very important because of the existing potential disruption of systems which is associated with risk to life and global catastrophes.

Recently, there are a lot of hierarchical and distributed approaches applicable to MPC technique. Let's concern with the part of the spectrum of these approaches.

As an example for hierarchical structures the *Price Method* [2] (for stationary and quasi-stationary processes) and the *Interaction Balance Method* [2] (for dynamic processes) can be used. Both are using Lagrange multipliers as a sub-coordination variable.

It is also known the *Three-level Optimization Method* by Tamura [2]. In comparison with the *Interaction Balance Method* this method used additional time decomposition on the lower level. It means that each subsystem is split into pieces; each of them is responsible for a certain time k.

In 2011 a new method was proposed by Doan et al. It is called the *Hierarchical Primal Feasible Dual Gradient Ascent Approach* [3]. It is applicable to solve large-scale MPC problems with coupling dynamics and constraints, which guarantees primal feasible solution even after a finite number of iterations.

For additional information about the hierarchical control methods mentioned above refer to [4] – [6].

Many approaches exist for the distributed techniques, such as the *Distributed Han's Algorithm* [9] and its improved [10] and modified [11] version. The original version of Han's and Lou's algorithm is described in [8]. The *Distributed Han's Algorithm* is based on the idea of *the Projected Gradient Method* [7].

In 2011 The *Sensitivity-Driven Distributed MPC Approach* [12] was suggested by Scheu and Marquardt. This new technique uses a distributed

control optimization algorithm, based on the sensitivity-driven coordination mechanism. Coordination and, consequently, the overall optimality are achieved using a linear approximation of the objective function of the neighboring controllers in the objective function of each local controller.

In 2001 Jia and Krogh [13] proposed *the Stability-Constrained MPC Algorithm* (DMPC-SC). In this mechanism controllers coordinate with each other by exchanging their predictions. There is no central coordinator. The controllers exchange information about their measurements and predictions and incorporate this information in their local computations.

In this research the comparison analysis of hierarchical and distributed optimization for MPC is presented. For the comparison the hierarchical *Interaction Balance Method* and the distributed approach, namely the *Distributed Han's Algorithm* have been selected, both applicable within MPC.

The goal is to recognize which control technique fits best for controlling the water supply or other similar systems.

This paper is organized as follows. In the2nd part the MPC structure will be considered. In the 3rd and 4th part a hierarchical and a distributed technique for the MPC will be performed, respectively. The application of selected hierarchical and distributed methods to the MPC applied to the numerical example is in the 5th part. In the 6th section the comparison analysis and conclusions will be introduced. The summary concludes the thesis.

2. Model predictive control

Currently, the scope of practical MPC methods applications has expanded, encompassing a variety of technological processes in the chemical industry and civil construction works, in consumer goods manufacturing, in water supply and sewer systems, in aerospace research and advanced energy systems, etc.

The main advantage of MPC approach in determining its successful use in the practical construction and operation of control systems is the relative simplicity of the basic scheme feedback's formation, combined with high adaptive properties. The latter allows managing some multi-dimensional and multi-object systems with a complex structure consisting of non-linearity, optimizing processes in real time within the constraints imposed on the manipulated and controlled variables to take into account the uncertainty in the assignment of objects, state restrictions, and disturbances. In addition, it is worth noting the possibility of improving the quality of the process during its execution.

Model predictive control is an optimal control strategy based on numerical optimization. Future control inputs and future plant responses are predicted using a system model. The control and state variables are calculating at the prediction interval with respect to an associated objective function. Model predictive control (MPC), also referred to as moving horizon control or receding horizon control, has become an attractive feedback strategy.

Typical goals of MPC are:

- ❖ Reduce variability by eliminating disturbances in the key variables;
- ❖ Improving the process, plant performance, product quality;
- ❖ A stable and safe operation.

Despite being very simple to design and implement, MPC algorithms can provide large-scale systems (LSS) with many controls. MPC technique represents a systematic method of dealing with constraints on controls and states. These constraints are presented in all control engineering applications and explain limitation of plants by their physical, economical, or safety side. In MPC these constraints are accounted by solving a constrained optimization problem in real-time to determine the optimal predicted inputs.

MPC-approach represents a following control scheme for dynamic objects on the basis of the feedback:

1. Considering a mathematical model of the object, the initial conditions for which is the current state. For a given control strategy performed integration of the equations of the model that predicts the behavior of the object in a finite period of time (prediction horizon).
2. Optimization process is proceeding. The goal is to lead the current variables to their desired conditions on the prediction horizon. Optimization takes into account the whole range of restrictions imposed on the control and state variables.
3. In the calculation step, constituting a fixed fraction of the prediction horizon, the optimal control is realized and we can obtain the actual state of the object at the end of the horizon.
4. The prediction horizon is shifted one step forward, and steps 1 - 3 are repeated in the action sequences.

This scheme can be combined with the prior conduct of the identification of the model equations used for prediction. Currently, MPC-approach is in the stage of intensive development, as evidenced by the extensive bibliography published in recent years of scientific papers on the subject.

In the researched literature MPC is formulated almost always in the state space. Let the model (2.1) and (2.2) of the plant be described by the linear difference equations:

$$\boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot \boldsymbol{u}(k), \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad (2.1)$$

$$\boldsymbol{y}(k) = \boldsymbol{C} \cdot \boldsymbol{x}(k), \quad (2.2)$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$,$\boldsymbol{u}(k) \in \mathbb{R}^m$, and $\boldsymbol{y}(k) \in \mathbb{R}^p$ denote the state, control input, and output vector, respectively, k is the discrete time. For the states k relates to the time point. Term time interval k applied for controls and disturbances.

MPC is a multivariable control algorithm that uses an internal dynamic model of the process, and an optimization cost function J over the receding prediction horizon, to calculate the optimal control moves.

The predictive controls and states is computed by minimizing a predicted objective cost (2.3), which is defined in terms of the predicted sequences $\boldsymbol{u}$,$\boldsymbol{x}$, where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weighting matrices, and N is the number of subsystems.

$$J(k) = \sum_{i=0}^{N} \boldsymbol{x}^T(k) \cdot \boldsymbol{Q} \cdot \boldsymbol{x}(k) + \boldsymbol{u}^T(k) \cdot \boldsymbol{R} \cdot \boldsymbol{u}(k) \qquad (2.3)$$

The MPC approach has many advantages and benefits such as:

- ❖ Explicitly handles constraints;
- ❖ Development time is much shorter than for competing advanced control methods;
- ❖ Changing model or specification does not require complete redesign, sometimes can be done on the fly;
- ❖ Explicit use of a model;
- ❖ Superior for processes with large number of manipulated and controlled variables.

Control methods based on the MPC concept have been widely used in industry and have been examined by the scientific community. At present it is the most widely used of all the modern control techniques in the industrial sector. The reason for such popularity is the ability to design the control system that can operates without the intervention of an expert for long periods of time.

A block diagram for the standard MPC implementation is shown in Figure 1.

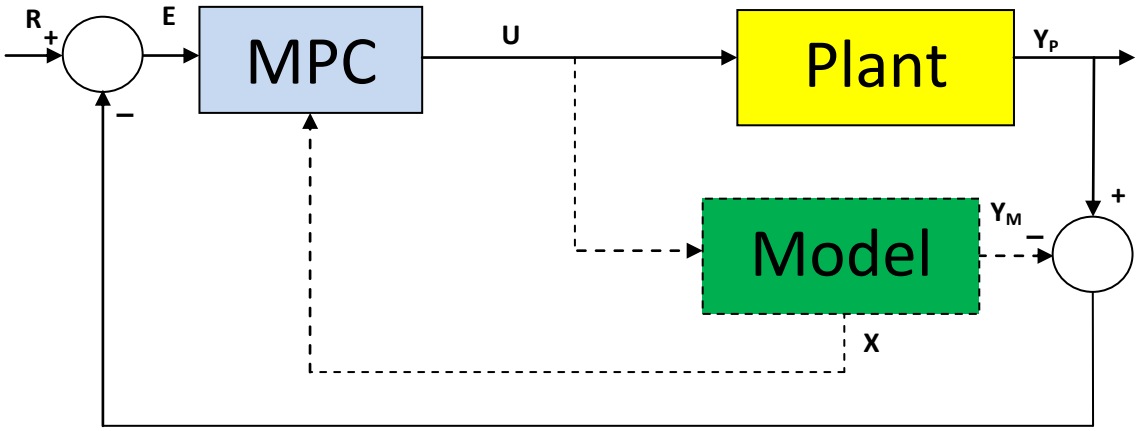

Figure 1: Implementation of MPC to plant

As shown in the figure, a process model is used in parallel to the plant. MPC uses a dynamic model of the process in order to predict the control and/or state variables. The predicted controlled variable is fed back to the controller where it is used in an on-line optimization procedure, which minimizes an appropriate objective function to determine the manipulated variable. The controller output is implemented in real time and then the procedure is repeated every sampling time with actual process data. The difference between the plant measurement Y_P and the model output Y_M is

also fed to the controller to eliminate steady state offset. Usually the cost function depends on the quadratic error between the reference state variable and the calculated state variable within limited time horizon.

In Fig. 2 state $\boldsymbol{x}$ and control signal $\boldsymbol{u}$ will be presented below the graph of the system for a better understanding of the control techniques. It means e. g., there are states and controls for the discrete time $k = 0,1,2,3$ and the goal is to predict these variables for the next whole period for $k = 4,5,6$.

It should be taken into account that the disturbances are acting in the system and the system states $\boldsymbol{x}$ at the discrete time instants may be different from the calculated state $\boldsymbol{x_c}$. The task of the MPC techniques is to bring the system to the required state $\boldsymbol{x_p}$ in the next period using a control signal $\boldsymbol{u}$. With application the optimization method the new control steps can be calculated and attached to the system.

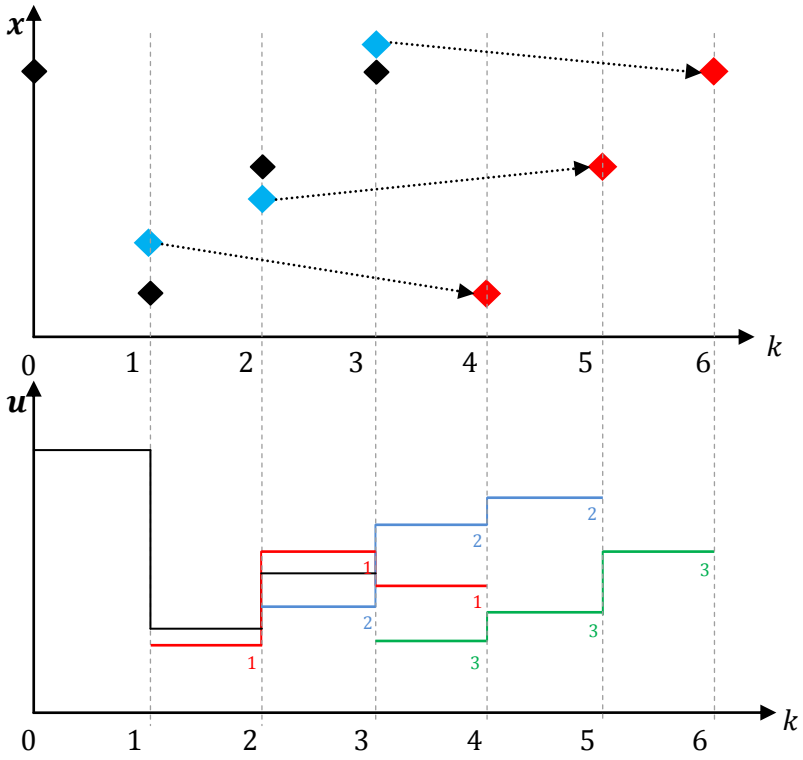

Figure 2: Example of the behavior of the system with MPC.

Legend of Fig. 2:

- Graph above: black square – calculated states $\boldsymbol{x_c}$, so called off-line solution; blue square – real states $\boldsymbol{x}$, red square – desired state $\boldsymbol{x_p}$.
- Graph below: black line – control signal without disturbances, red, blue, green line – control sequence under influence of the disturbances with number, which explains prediction step.

In next two chapters the hierarchical and the distributed methods selected applicable to the MPC will be considered.

3. Hierarchical optimization methods

3.1 General description of hierarchical optimization methods

In this section a general theory for the optimization of two-level hierarchical systems will be introduced. This theory can be applied to a wide range of applications. Examples can be organization theory, economic networks, industrial plants, etc.

The development and analysis of complex systems often require the system's division into small units, called subsystems. The structure of common systems resulting from the interconnections of subsystems can be very complicated and sophisticated. One of the most general compositions is the hierarchical structure. The layout of the structure is vertical.

For example, a diagram of a standard two-level hierarchical system is shown in Figure 3, where two levels can be found.

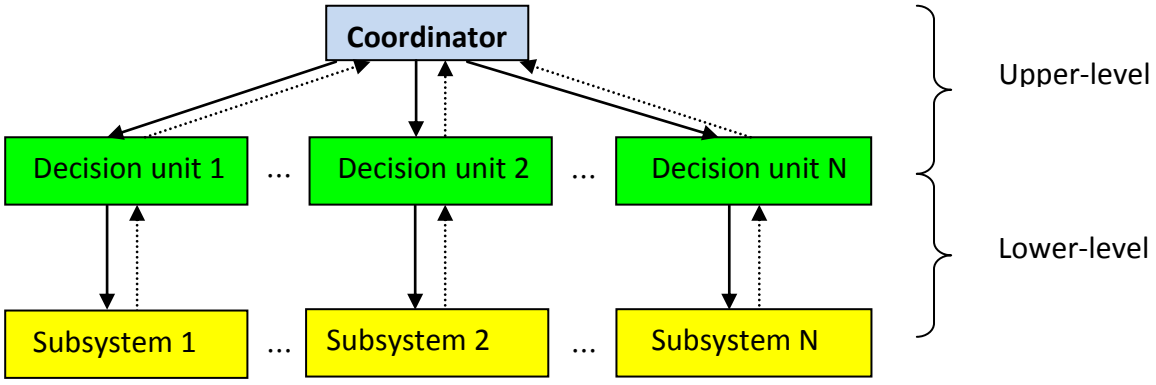

Figure 3: A diagram of a standard two-level hierarchical system.

The levels are called the lower level and the upper level. The lower level consists of the process level, where the process level has been divided into N subsystems. The subsystems are connected to each other because there are information flows between these subsystems. Each subsystem has its own decision unit, which controls the behavior of the subsystem so that the objectives of this particular subsystem will be met. However, quite often targets of subsystems are in conflict with each other, and as a result, the overall system performance is reduced. Hence an upper-level decision unit or a coordinator has to be included in the scheme, and the objective of this decision unit is to coordinate the decision making of the subsystems therefore the overall performance of the system will be optimized. The coordinator receives information from the subsystems thereby it can supervise the performance of the overall system, so that an overall harmony is achieved.

There is a problem that not all systems are amenable to establish a hierarchical structure. The following items can be highlighted as a condition for the formation of such structuring:

- The whole system is suited to be splitted into subsystems;
- Set of allowable controls must be separated into subsets;
- Increase/decrease in partial objective functions leads to increase/decrease in the overall objective function;
- A subsystem is only responsible for its own task and does not require any information about the objectives of the overall system.

When choosing a particular control technology, the procedure should be guided by the following instructions:

- Choice of the coordinate variables;
- Influence on the process description;
- Application conditions of the control;
- Properties of application scope.

In management of LSS two-level methods are very attractive, because the control problem can be solved in several subsystems. This leads to more comfortable composition of a LSS and fast computation. In the optimal control problem for LSS the problem is to minimize an objective function subject to the systems dynamics, restrictions, equations, and inequalities. Using a two-level method the system should be divided into several interconnected subsystems and the cost function should be divided into several objective functions for each subsystem. Minimization or otherwise maximization of these cost functions for each subsystem leads to several sub-problems.

In the next subsections the several hierarchical structures for processes will be presented.

3.2 The principle of price coordination

This approach [2] is a method for coordinating the sub-problem solutions in plant decomposition, in which Lagrange multipliers enter into the subsystem cost functions as shadow prices, and these are adjusted by the coordinator in an iterative procedure which culminates in the satisfaction of the subsystem coupling relationships (equations).

The *Price Coordination* mechanism represents itself as the flexible approach to modeling, optimization and control of complex systems. This follows the natural role played by prices to achieve the balance between the demand and the supply. The optimization problem for a subsystem is presented by (3.2.1a).

$$\min_{x_i, v_i} J_i(x_i, v_i) \qquad (3.2.1a)$$

The method allows various problem formulations and price adjustment strategies. Problem formulation may either result from a simple decomposition of a large-scale optimization problem. For the method description, we consider a simple system [2]. In order to apply the *Price Method*, the Lagrangian function $L_i(x_i, v_i, \boldsymbol{\lambda})$ for a subsystem has to be written (3.2.1b).

$$L_i(x_i, v_i, \boldsymbol{\lambda}) = J_i(x_i, v_i) + \lambda_i \cdot v_i - \lambda_j \cdot h_i(x_i, v_i), i, j = 1,2, i \neq j \quad (3.2.1b)$$

In Figure 4 the scheme of the *Price Method* is presented.

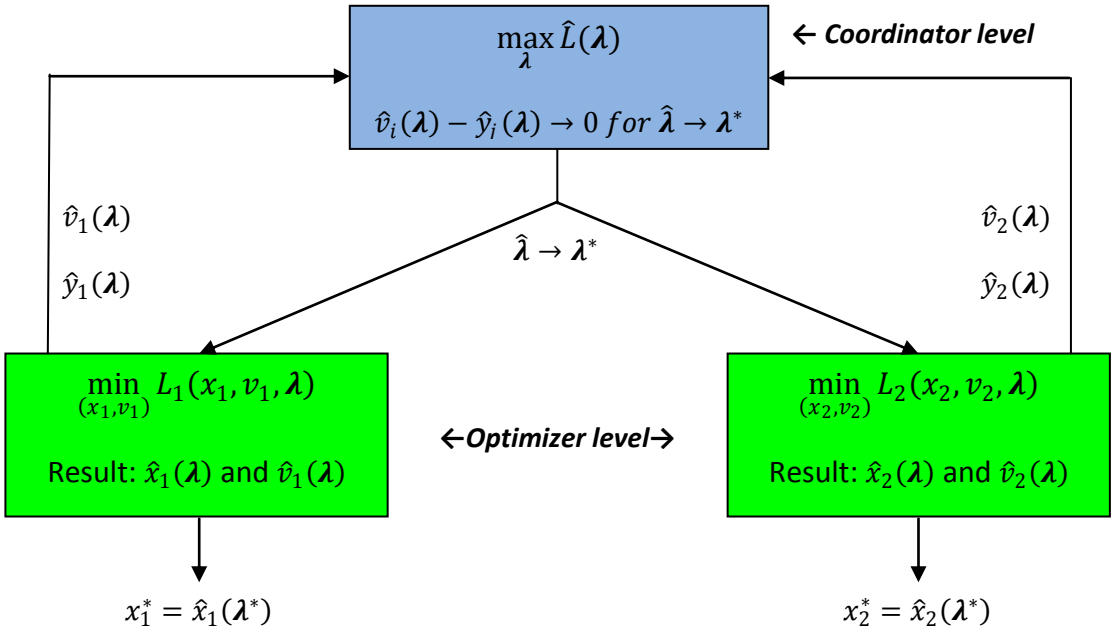

Figure 4: The Price Coordination mechanism

*Comments for Fig. 4: Symbols with ˆ mean conditional optimal value, and symbols with * mean optimal value*

L_i and h_i depend only from values of subsystem i. In this method the optimality condition is the saddle point condition (3.2.2 \ 3.2.3).

$$J^* = \max_{\boldsymbol{\lambda}} \hat{L}(\boldsymbol{\lambda}) \qquad (3.2.2)$$

$$J^* = \max_{\boldsymbol{\lambda}} \left\{ \min_{(x_1, v_1)} L_1(x_1, v_1, \boldsymbol{\lambda}) + \min_{(x_2, v_2)} L_2(x_2, v_2, \boldsymbol{\lambda}) \right\} \qquad (3.2.3)$$

Optimizer determines the conditional optimal control $\hat{x}_i(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda}$ at specified prices λ_i for the demand v_i as well as prices λ_j for the supply y_i. Only after completion of the optimization optimal control variables can be given to the real process. During the optimization couplings are not fulfilled and the control strategy is not allowed.

The dimension of the coordinator optimization problem equals $\dim \boldsymbol{\lambda}$, and the dimension of subsystem optimization problems i equals $\dim x_i + \dim v_i$.

The scheme in the Figure 4 is suitable for static models in contrast to the *Interaction-Balance Method* (IBM), which will be considered in the next sub-section and represents itself the modification of the *Price Method*, the IBM fits for dynamical systems.

3.3 The interaction balance method

The IBM approach [2] has a more complicated structure as The *Price Coordination* method. It is expressed in the fact that now a linear-quadratic problems (linear or linearized system description) and quadratic objective functional are applied. Moreover, in a hierarchical system the task of the coordinator is not to control but to coordinate and to manage the functioning of the lower subsystems so as to meet the requirement of the whole system at lowest performance index. Using these principles for control, these systems are decomposed into several subsystems with interaction inputs from each other. Each subsystem solves its own partial optimization problem and a coordinator manages these low level optimizers to solve the global optimization problem. Applying the IBM, the coordinator does not change the objective function. It defines the violation of the constraint equations and determines the new values of coordination variables. The coordinator maximizes the dual function, reduces the step size and thereby coupling error declines.

The optimization problem can be formulated in this way (3.3.1 – 3.3.10).

- The system description is the state differential equation (3.3.1), with $\boldsymbol{A}_i$ – state matrix, $\boldsymbol{B}_i$ – control matrix, $\boldsymbol{C}_i$ – coupling matrix, $\boldsymbol{x}_i(t)$ – state

vector, $\boldsymbol{u}_i(t)$ – control vector, $\boldsymbol{v}_i(t)$ – input/coupling vector, index i – corresponds to i^{th} subsystem, N – number of subsystems:

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{A}_i \cdot \boldsymbol{x}_i(t) + \boldsymbol{B}_i \cdot \boldsymbol{u}_i(t) + \boldsymbol{C}_i \cdot \boldsymbol{v}_i(t),$$
$$\boldsymbol{x}_i(0) = \boldsymbol{x}_{i,0}, i = 1,2, \dots, N. \quad (3.3.1)$$

- Coupling equation (3.3.2), where $\boldsymbol{K}_{ij}$ – coupling matrix.

$$\boldsymbol{v}_i(t) = \sum_{j=1}^{N} \boldsymbol{K}_{ij} \cdot x_j(t) \quad (3.3.2)$$

- Objective functional (3.3.6), with

$$\|x_i(t)\|^2_{Q_i} = x_i^T(t) \cdot Q_i \cdot x_i(t); \quad (3.3.3)$$
$$\|u_i(t)\|^2_{R_i} = u_i^T(t) \cdot R_i \cdot u_i(t); \quad (3.3.4)$$
$$\|v_i(t)\|^2_{S_i} = v_i^T(t) \cdot S_i \cdot v_i(t); \quad (3.3.5)$$

$$J\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{v}_i(t)\big) =$$
$$= \sum_{i=1}^{N} \left[\frac{1}{2} \cdot \big\|\boldsymbol{x}_i(t_f)\big\|^2_{\bar{Q}_i} + \frac{1}{2} \right.$$
$$\left. \cdot \int_0^{t_f} \big(\|x_i(t)\|^2_{Q_i} + \|u_i(t)\|^2_{R_i} + \|v_i(t)\|^2_{S_i}\big) dt \right]. \quad (3.3.6)$$

- Dual function (3.3.7) subject to (3.3.1); L_i – Lagrangian function (3.3.8), $\boldsymbol{\lambda}$ - Lagrange multiplier.

$$\Phi(\boldsymbol{\lambda}) = \min_{(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v})} L_i(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{v}_i(t), \boldsymbol{\lambda}) \quad (3.3.7)$$

$$L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} L_i(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{v}_i(t), \boldsymbol{\lambda})$$
$$= \sum_{i=1}^{N} \left\{ \frac{1}{2} \cdot \big\|\boldsymbol{x}_i(t_f)\big\|^2_{\bar{Q}_i} \right.$$
$$+ \int_0^{t_f} \left[\frac{1}{2} \cdot \|x_i(t)\|^2_{Q_i} + \frac{1}{2} \cdot \|u_i(t)\|^2_{R_i} + \frac{1}{2} \cdot \|v_i(t)\|^2_{S_i} + \boldsymbol{\lambda}^T \right.$$
$$\left. \left. \cdot \Big(\boldsymbol{v}_i(t) - \sum_{j=1}^{N} K_{ij} \cdot x_j(t)\Big) \right] dt \right\} \quad (3.3.8)$$

- For the description of coupling error we can construct the gradient (3.3.9) of Lagrangian function with respect to $\boldsymbol{\lambda}$.

$$\nabla_{\boldsymbol{\lambda}}\Phi(\boldsymbol{\lambda}) = \boldsymbol{v}_i(t) - \sum_{j=1}^{N} K_{ij} \cdot x_j(t) = e_i, i = 1,2,\dots,N \qquad (3.3.9)$$

- ❖ The algorithm can cyclically repeat until the error is zero or near zero. In each iteration the Lagrange multiplier is exposed to improvement. The states in next iteration can be calculated by the formula (3.3.10), where $\boldsymbol{e}^k(t)$ – total error vector (search direction), α^k – step size.

$$\lambda^{*,k+1}(t) = \lambda^{*,k}(t) + \alpha^k \cdot \boldsymbol{e}^k(t), 0 \le t \le t_f \qquad (3.3.10)$$

This method is widely used in hierarchical control systems. This approach has the following advantages (+) and disadvantages (-):

+ In principal, it is possible to consider the inequality constraints.
+ Convergence against optimum.
+ Parallelization of the problems at the lower level.
- Choice of step size α on the coordination level (it can iteratively be changed or be constant).
- There is no admissible control during optimization.
- Inclusion of $\frac{1}{2} \cdot \|v_i(t)\|_{S_i}^2$ has often a little practical importance.

4. Distributed optimization methods

4.1 General description of distributed optimization methods

Distributed optimization algorithms can be useful from several points of view. These methods are generally used to increase speed and numerical efficiency. Distributed MPC methods are very popular in the industry. Key benefit of MPC is that it can hold tight restrictions in the inputs, states, and outputs of the control system. For LSS, centralized MPC is considered often as an impracticable and unsuitable because of its computational and information exchange limits. In order to address these restrictions, distributed MPC (DMPC) should be used.

In distributed control architectures [6], like the simple example shown in Figure 5, it is assumed that some information is transmitted (and received) among the local regulators, so that each one of them has some knowledge of the behavior of others. When the local controllers are designed with MPC, the information transmitted typically consists of the future predicted control or state variables computed locally, so that any local regulator can predict the interaction effects over the considered prediction horizon.

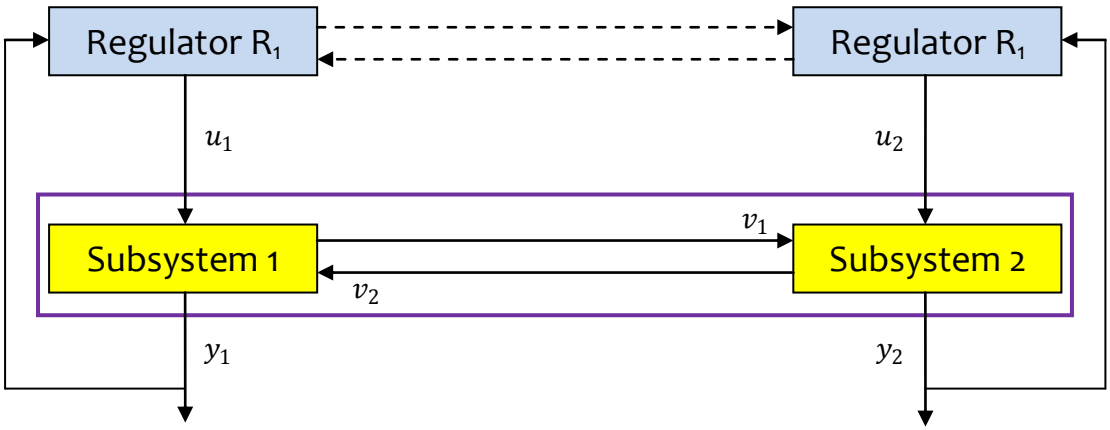

Figure 5: General scheme of distributed optimization methods

In a wide range of distributed algorithm MPC, proposed in the literature [11] – [14], the classification can be done depending on the topology of the system communication. In particular, classification in the following cases has to be considered:

- ❖ information is transmitted/received to/from any local regulator to all the others (fully connected algorithm);

- ❖ information is transmitted/received to/from any local regulator to a given subset of the others (partially connected algorithm).

A partially connected system structure can be useful in the case of LSS, which is made of a large number of loosely coupled subsystems, i.e. where the coupling matrix has a sparse structure. The exchange of information among local regulators can be made according to different protocols:

- ❖ information is transmitted by the local regulators only once within each sampling time (non-iterative algorithms);
- ❖ information can be transmitted by the local regulators many times within the sampling time (iterative algorithms).

Obviously, the volume of information, available to the local controllers with iterative algorithms, is higher so that the overall iterative procedure can be set to achieve a global consensus on actions to be taken during the sampling interval. In this regard, however, further classification should be considered:

- ❖ distributed algorithms, where each local regulator minimizes a local performance index (independent algorithms);
- ❖ distributed algorithms, where each local regulator minimizes a global cost function (cooperating algorithms).

4.2 The projected gradient method

In this subsection Rosen's gradient projection method [7] will be considered. It is based on projecting the search direction into subspace tangent to the active constraints. Only the case of linear constraints will be considered. The constrained problem is defined as follows:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.2.1}$$

$$\text{s. t. } g_j(\boldsymbol{x}) = \sum_{i=1}^{n} a_{ij} \cdot x_i - b_j \geq 0\,, j = 1, \dots, n_g \tag{4.2.2}$$

The equation (4.2.2) can be rewritten in vector form (4.2.3).

$$g_j = \boldsymbol{a}_j^T \cdot \boldsymbol{x} - b_j \geq 0 \tag{4.2.3}$$

If only the r active constraints ($j \in I_A$) have been selected, the constraint equations will be as

$$\boldsymbol{g}_a = \boldsymbol{N}^T \cdot \boldsymbol{x} - \boldsymbol{b} = 0, \tag{4.2.4}$$

where $\boldsymbol{g}_a$ is the vector of active constraints and the columns of the matrix $\boldsymbol{N} = \frac{\partial g_j}{\partial x_i}, j \in I_A, i = 1 \ldots n_g$ are the gradients of these constraints. The basic assumption of the gradient projection method is that $\boldsymbol{x}$ lies in the subspace tangent to the active constraints. If

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha \cdot \boldsymbol{s}, \tag{4.2.5}$$

and both x_i and x_{i+1} satisfy equation (4.2.4), then

$$\boldsymbol{N}^T \cdot \boldsymbol{s} = 0. \tag{4.2.6}$$

If the steepest descent direction satisfying (4.2.6), the problem can be posed as

$$\min \boldsymbol{s}^T \cdot \nabla f \tag{4.2.7a}$$

$$\text{s. t. } \boldsymbol{N}^T \cdot \boldsymbol{s} = 0 \text{ and } \boldsymbol{s}^T \cdot \boldsymbol{s} = 1. \tag{4.2.7b}$$

I. e., it is required to find the direction with the negative directional [7] derivative which satisfies (4.2.6). Lagrange multipliers $\boldsymbol{\lambda}$ and μ were used to the form the Lagrangian

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{\lambda}, \mu) = \boldsymbol{s}^T \cdot \nabla f - \boldsymbol{s}^T \cdot N \cdot \boldsymbol{\lambda} - \mu \cdot (\boldsymbol{s}^T \cdot \boldsymbol{s} - 1). \tag{4.2.8}$$

The condition for $\mathcal{L}$ to be stationary is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}} = \nabla f - N \cdot \boldsymbol{\lambda} - 2 \cdot \mu \cdot \boldsymbol{s} = 0. \tag{4.2.9}$$

Premultiplying (4.2.9) by $\boldsymbol{N}^T$ and using (4.2.6) it is obtained

$$\boldsymbol{\lambda} = (\boldsymbol{N}^T \cdot \boldsymbol{N})^{-1} \cdot \boldsymbol{N}^T \cdot \nabla f. \tag{4.2.10}$$

Use equations (4.2.9) and (4.2.10) and get (4.2.11)

$$\boldsymbol{s} = \frac{1}{2 \cdot \mu} \cdot [I - \boldsymbol{N} \cdot (\boldsymbol{N}^T \cdot \boldsymbol{N})^{-1} \cdot \boldsymbol{N}^T] \cdot \nabla f = \frac{1}{2 \cdot \mu} \cdot \boldsymbol{P} \cdot \nabla f. \tag{4.2.11}$$

$\boldsymbol{P}$ is the projection matrix. The term $\frac{1}{2 \cdot \mu}$ is not significant because $\boldsymbol{s}$ defines only the search direction, thus in general applied $\boldsymbol{s} = \boldsymbol{P} \cdot \nabla f$. To show that $\boldsymbol{P}$ indeed has the projection property. The proof that if $\boldsymbol{w}$ is an arbitrary vector, and then $\boldsymbol{P} \cdot \boldsymbol{w}$ is in the subspace tangent to the active constraints, that is $\boldsymbol{P} \cdot \boldsymbol{w}$ satisfies (4.2.12) is required, with $\boldsymbol{P}$ is defined by (4.2.13).

$$\boldsymbol{N}^T \cdot \boldsymbol{P} \cdot \boldsymbol{w} = 0, \tag{4.2.12}$$

$$\boldsymbol{P} = \boldsymbol{Q}_2^T \cdot \boldsymbol{Q}_2, \tag{4.2.13}$$

where $\boldsymbol{Q}_2$ consists of the last $n - r$ rows of $\boldsymbol{Q}$.

The Lagrange multipliers can be calculated by (4.2.10). If all the components of $\boldsymbol{\lambda}$ are nonnegative, the Karush-Kuhn-Tucker (KKT) conditions are indeed satisfied and the optimization can be terminated. If some of the Lagrange multipliers are negative, it is an indication that while no progress is possible with the current set of active constraints, it may be possible to proceed by removing some of the constraints associated with negative Lagrange multipliers. A common strategy is to remove the constraint associated with the negative Lagrange multiplier and to repeat the calculation of $\boldsymbol{P}$ and $\boldsymbol{s}$. If $\boldsymbol{s}$ is now non-zero, a one-dimensional search may be started. If $\boldsymbol{s}$ remains zero and there are still negative multipliers, another constraint is removed until all Lagrange multipliers become positive and the KKT conditions are satisfied.

After a search direction has been determined, a one dimensional search must be carried out to determine the value of α in equation (4.2.5). Unlike the unconstrained case, there is an upper limit on α set by the inactive constraints. As α increases, some of them may become active and are then violated. Substituting (4.2.5) into (4.2.3) it is obtained

$$g_j = \boldsymbol{a}_j^T \cdot (\boldsymbol{x}_i + \alpha \cdot \boldsymbol{s}) - b_j \geq 0 \tag{4.2.14a}$$

or

$$\alpha \leq -\frac{\boldsymbol{a}_j^T \cdot \boldsymbol{x}_i - b_j}{\boldsymbol{a}_j^T \cdot \boldsymbol{s}} = \frac{-g_j(\boldsymbol{x}_i)}{\boldsymbol{a}_j^T \cdot \boldsymbol{s}}. \tag{4.2.14b}$$

Equation (4.2.14b) is valid if $\boldsymbol{a}_j^T \cdot \boldsymbol{s} < 0$. Otherwise, there is no upper limit on α due to the j^{th} constraint. From (4.2.14b) there is a different α, say α_j for each constraint. The upper limit on α is the minimum from all $\alpha_j > 0, j \in I_A$:

$$\bar{\alpha} = \min_{\alpha_j > 0, j \in I_A} \alpha_j \,. \tag{4.2.15}$$

At the end of the move, new constraints may become active, so that the set of active constraints may need to be updated before the next move

is undertaken. The version of the gradient projection method presented so far is an extension of the steepest descent method. Like the steepest descent method, it may have a slow convergence rate.

In the next subsection the original Han's method will be presented, which uses the idea of the gradient projection.

4.3 Han's method for convex programs

In this section Han's method for convex programs which was introduced in 1988 by S.-P. Han and G. Lou [8] will be explained.

The optimization problem can be represented by (4.3.1)

$$\min_{\boldsymbol{x}\in C=C_1\cap\cdots C_m} q(\boldsymbol{x}) \tag{4.3.1}$$

where the function q is uniformly convex and differentiable on $\mathbb{R}^n$ and C_i are closed convex sets and $C \neq \emptyset$. In particular, the method can be used for tackling definite quadratic problems. This approach is an iterative procedure. The main computation in each iteration is to solve m subproblems of the following form (4.3.2)

$$\min_{\boldsymbol{z}\in C_i} \frac{1}{2}\cdot\|\boldsymbol{z}-\boldsymbol{w}_i\|^2 \tag{4.3.2}$$

with vector $\boldsymbol{w}_i$ varying iteratively for each set C_i. The main feature here is that m subproblems are independent of each other and can be solved simultaneously and, therefore, the method is suitable for parallel computations.

The quadratic optimization problem of the following form (4.3.3) is considered subject to (4.3.3a) and (4.3.3b) with $s = n_{eq} + n_{ineq}$, where n_{eq} and n_{ineq} are numbers of equality and inequality constraints, respectively, and $\boldsymbol{H}$ being a positive definite matrix.

$$\min_{\boldsymbol{x}} \boldsymbol{x}^T\cdot\boldsymbol{H}\cdot\boldsymbol{x} \tag{4.3.3}$$

$$\boldsymbol{a}_l^T\cdot\boldsymbol{x} = b_l, l = 1,\dots,n_{eq} \tag{4.3.3a}$$

$$\boldsymbol{a}_l^T\cdot\boldsymbol{x} \leq b_l, l = n_{eq}+1,\dots,s \tag{4.3.3b}$$

For the usage of this method the conjugate function of function $q(\boldsymbol{x})$ is required. This function denotes as $q^*(\boldsymbol{y})$ and by the formula (4.3.4) we can calculate it.

$$q^*(\boldsymbol{y}) = \sup_{\boldsymbol{x}\in\mathbb{R}^{n_x}} (\boldsymbol{y}^T \cdot \boldsymbol{x} - q(\boldsymbol{x})) \tag{4.3.4}$$

As it was described previously, a quadratic objective function is used $q(\boldsymbol{x}) = \boldsymbol{x}^T \cdot \boldsymbol{H} \cdot \boldsymbol{x}$ and it means that the conjugate function can be calculated by (4.3.5). The conjugate function $q^*(\boldsymbol{y})$ is also convex and differentiable on $\mathbb{R}^n$ as the original function $q(\boldsymbol{x})$.

$$q^*(\boldsymbol{y}) = \boldsymbol{y}^T \cdot \boldsymbol{H}^{-1} \cdot \boldsymbol{y} \tag{4.3.5}$$

Also it is defined p as iteration counter of the algorithm and a superscript (p) to denote the values of variables computed at iteration p. The parameter α has to be chosen as a sufficiently large number which can be calculated by (4.3.4), with s is the number of constraints and ω is one half of the smallest eigenvalue of $\boldsymbol{H}$.

$$\alpha = \frac{s}{\omega} \tag{4.3.4}$$

The algorithm of Han's method is represented below.

Initialization with $p = 0, \boldsymbol{y}^{(0)} = \boldsymbol{y}_1^{(0)} = \cdots = \boldsymbol{y}_s^{(0)} = \boldsymbol{0}$, with $\boldsymbol{y}^{(0)}, \boldsymbol{y}_l^{(0)} \in \mathbb{R}^{n_x}, l = 1, \dots, s,$ and $\boldsymbol{x}^{(0)} = \nabla q^*\left(\boldsymbol{y}^{(0)}\right)$. For the $p = 1,2, \dots$ we perform the following computations:

1. For $l = 1, \dots, s$ find $\boldsymbol{z}_l^{(p)}$ that solves (4.3.5)

$$\min_{\boldsymbol{z}\in C_l} \frac{1}{2} \cdot \left\| \boldsymbol{z} + \alpha \cdot \boldsymbol{y}_l^{(p-1)} - \boldsymbol{x}^{(p-1)} \right\|_2^2 \tag{4.3.5}$$

2. For $l = 1, \dots, s$ assign (4.3.6)

$$\boldsymbol{y}_l^{(p)} = \boldsymbol{y}_l^{(p-1)} + \frac{1}{\alpha} \cdot \left(\boldsymbol{z}_l^{(p)} - \boldsymbol{x}^{(p-1)} \right) \tag{4.3.6}$$

3. Set

$$\boldsymbol{y}^{(p)} = \boldsymbol{y}_1^{(p)} + \cdots + \boldsymbol{y}_s^{(p)} \tag{4.3.7}$$

4. Compute

$$\boldsymbol{x}^{(p)} = \boldsymbol{\nabla} q^*\left(\boldsymbol{y}^{(p)}\right) \tag{4.3.8}$$

Han and Lou (1988) showed that $\left\|\boldsymbol{y}^{(p)}-\boldsymbol{y}^{(p-1)}\right\|_2 \to 0$ and $\left\|\boldsymbol{x}^{(p)}-\right.$ $\boldsymbol{x}p-12\to 0$ as $p\to\infty$. They also showed that their algorithm converges to the global optimum if $q(\boldsymbol{x})$ is uniformly convex and differentiable on$\mathbb{R}^{n_x}$.

In the fourth step of the algorithm the gradient of conjugate function has to be calculated. It can be proceed this by the formula (4.3.9) only in case, if $q(\boldsymbol{x})$ is a quadratic function of the form (4.3.3).

$$\nabla q^*\left(\boldsymbol{y}^{(p)}\right)=\boldsymbol{H}^{-1}\cdot\boldsymbol{y} \tag{4.3.9}$$

However, in Han's algorithm for the quadratic optimization problems, it is not necessary to compute $\boldsymbol{z}^{(p)}$, and $\boldsymbol{y}^{(p)}$can be eliminated using (4.3.10),

$$\boldsymbol{y}_l^{(p)}=-\frac{\gamma_l^{(p)}}{\alpha\cdot\boldsymbol{a}_l^T\cdot\boldsymbol{a}_l}\cdot\boldsymbol{a}_l \tag{4.3.10}$$

which leads to the following simplified approach.

Initialization with $p=0$, $\gamma_1^{(p)}=\cdots=\gamma_s^{(p)}=0$. For $l=1,\ldots,s$ compute $\boldsymbol{c}_l$ by formula (4.3.11).

$$\boldsymbol{c}_l=\frac{1}{\alpha\cdot\boldsymbol{a}_l^T\cdot\boldsymbol{a}_l}\cdot\boldsymbol{H}^{-1}\cdot\boldsymbol{a}_l \tag{4.3.11}$$

For the steps $p=1,2,\ldots$ we perform the following computation.

1. For each l corresponding to an equality constraint $(1\le l\le n_{eq})$, compute (4.3.12)
$$\gamma_l^{(p)}=\boldsymbol{a}_l^T\cdot\boldsymbol{x}^{(p-1)}+\gamma_l^{(p-1)}-b_l \tag{4.3.12}.$$
For each l corresponding to an inequality constraint $(n_{eq}+1\le l\le s)$, compute (4.3.13)
$$\gamma_l^{(p)}=\max\left\{\boldsymbol{a}_l^T\cdot\boldsymbol{x}^{(p-1)}+\gamma_l^{(p-1)}-b_l,0\right\}. \tag{4.3.13}$$
2. Set (4.3.14)
$$\boldsymbol{x}^{(p)}=\sum_{l=1}^{s}\gamma_l^{(p)}\cdot\boldsymbol{c}_l \tag{4.3.14}$$

Note that Han's method splits up the computation into s parallel sub problems, with s the number of constraints. This algorithm is simple as original, but it still requires a global update scheme, and the parallel

problems still operate with the full-sized decision vector. Also it is worth noting that both methods have a poor convergence. Han and Lou remarked that the efficiency of the method certainly depends on how effectively we can solve the sub problems. In the next subsection the distributed Han's method [9] will be represented, that does not require global update communication.

4.4 The distributed version of Han's method

For this approach M local controllers attached to M subsystems were used. Each controller i computes with regard to a small set of constraints indexed by l. The same parameter α as for previous method has been chosen, and has been calculated variable $\boldsymbol{c}_l$. This variable is sparse as the corresponding $\boldsymbol{a}_l$. This parameter can be computed locally by a local controller with the knowledge of $\boldsymbol{a}_l$ and Hessian weight matrix $\boldsymbol{H}$. It is assumed that each local controller knows its own local dynamics. Hence, each local controller is in charge of updating variables of its system, and is updating some intermediate variables. The following two conditions have to be respected:

- Each constraint is taken into account by one and only one local controller even for a coupled constraint;
- A local controller can only be in charge of constraints that involve their own variables.

With respect to these conditions let denote L_i as the set of indices l that local controller i is in charge of and $L_{\mathcal{N}^i}$ as the set of indices l corresponding to the constraints that are taken into account by subsystem i or by any neighbor of i.

The distributed version of Han's algorithm is an iterative process in which four steps are executed. Two steps are communication steps between controllers, and two are computational steps. Before the method's description the main definitions have to be performed, which are used by the method.

The index matrix of subsystem: each subsystem i has a square matrix $\boldsymbol{J}^i \in \mathbb{R}^{n_x \times n_x}$ that is diagonal, with an entry on the diagonal being 1, if it corresponds to the position of a variable of subsystem i in the vector $\boldsymbol{x}$, and 0 otherwise. In short, this matrix is a selection matrix.

The vector $\boldsymbol{x}^{(p)|i} \in \mathbb{R}^{n_x}$ is a vector of the same size as $\boldsymbol{x}$, containing the values of i's variables computed at iteration p at the right positions, and zeros for the other entries. This vector called the self image of vector $\boldsymbol{x}^{(p)}$ made by subsystem i. The relation between the index matrix and the self image can be described by formula (4.4.1).

$$\boldsymbol{x}^{(p)|i} = J^i \cdot \boldsymbol{x}^{(p)} \tag{4.4.1}$$

For extension of the self image concept, the neighborhood image of subsystem i made from $\boldsymbol{x}$ is represented by (4.4.2). At the step p of the iteration, subsystem i constructs $\boldsymbol{x}^{(p)|\mathcal{N}^i}$ by putting the values of its neighbors' variables and its own variables to the right positions, and filling in zeros for the remaining slots of$\boldsymbol{x}$.

$$\boldsymbol{x}^{(p)|\mathcal{N}^i} = \sum_{\mathrm{j} \in \mathcal{N}^i} \boldsymbol{x}^{(p)|j} \tag{4.4.2}$$

The distributed Han's algorithm (DH) begins with initialization step with$p = 0$, $\boldsymbol{x}^{(0)|i} = \mathbf{0}$, $\boldsymbol{x}^{(0)} = \mathbf{0}$, and $\gamma_l^{(0)} = 0$ for $l = 1, \dots, s$.Next, for $p = 1,2, \dots$ the following actions are executed:

1. **Communication to get the updated main variables**
   Each controller i communicates with its neighbor $j \in \mathcal{N}^i$ to get updated values of their variables, contained in$\boldsymbol{x}^{(p-1)|i}$. Vice versa, i sends its updated variables in $\boldsymbol{x}^{(p-1)|i}$ to its neighbor as requested. After getting information from the neighbors, controller i constructs the neighborhood image $\boldsymbol{x}^{(p-1)|\mathcal{N}^i}$ using formula (4.4.2)
2. **Update intermediate variables $\boldsymbol{\gamma_l}$ in parallel**
   In this step, the local controllers update γ_l corresponding to each constraint l under their responsibility. To be more specific, each controller i updates γ_l for each $l \in L_i$ in the following manner:
   - If constraint$l$ is an equality constraint ($1 \le l \le n_{eq}$), then
   $$\gamma_l^{(p)} = \boldsymbol{a}_l^T \cdot \boldsymbol{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l \tag{4.4.3}$$
   - If constraint l is an inequality constraint ($n_{eq} + 1 \le l \le s$), then
   $$\gamma_l^{(p)} = \max\left\{\boldsymbol{a}_l^T \cdot \boldsymbol{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l, 0\right\} \tag{4.4.4}$$
3. **Communication to get the updated intermediate variables**
   Each controller i communicates with its neighbor to get updated $\gamma_l^{(p)}$ values that the neighbors just computed in step 2.

4. **Update main variables in parallel**

   Local controller i uses all $\gamma_l^{(p)}$ values that it has (by communications and those computed itself), to compute an assumed neighborhood image of $\boldsymbol{x}$ using (4.4.5).

$$\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} \cdot \boldsymbol{c}_l \tag{4.4.5}$$

   Note that $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$ has the same structure as $\boldsymbol{x}^{(p-1)|\mathcal{N}^i}$. However, it is not an exact update of the neighborhood image. Then controller i selects the value of its variables in $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$ to construct the new self image by (4.4.6).

$$\boldsymbol{x}^{(p)|i} = J^i \cdot \boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i} \tag{4.4.6}$$

5. **Check the termination criterion**

   In this step each local controller checks the local termination criterion. We propose the following termination using formula (4.4.7), where ε is a threshold.

$$\left\| \boldsymbol{x}^{(p)|i} - \boldsymbol{x}^{(p-1)|i} \right\|_2 \leq \varepsilon \tag{4.4.7}$$

   When all local controllers have converged, the algorithm stops and the local controller actions are implemented, otherwise the controllers proceed to step 1 to start a new iteration with $p = p + 1$.

At step 4 of the algorithm, each controller i makes an assumed neighborhood image $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$. The mechanism of this assumed neighborhoods image should be clarified. Controller i knows exactly only its own variables, while the variables of i's neighbors contained in $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$ are the assumption of controller i. Since i does not know the interaction between its neighbors and their other neighbors, thus their updates will be different from what i assumes for them. Therefore, i only extracts its variables from $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$. The real neighborhood image will be made in the next iteration after i receives updated values of its neighbor. This distributed algorithm uses the local update scheme. For the equivalence of this scheme to the simplified original Han's update scheme refer to [9].

The Hessian matrix $\boldsymbol{H}$ is assumed to be a positive definite matrix and optimization problem (4.3.3 / 4.3.3a / 4.3.3b) has a feasible solution. Based on both assumptions the distributed version of Han's algorithm converges to the centralized solution of the optimization problem (4.3.3 / 4.3.3a /

4.3.3b) at each sampling step. Assume that every sampling step the algorithm converges. Hence the DMPC scheme is recursively feasible and stable.

A main disadvantage of Han's method and its distributed version is the slow convergence rate, due to the fact that it is essentially a projection method to solve the dual problem of (4.3.3 / 4.3.3a / 4.3.3b). Furthermore, both approaches use zeros as initial guess of the solution. Thereby the improvement has to be done.

In the next subsections the improved version of DMPC scheme of Han's method [10] and its modification [11] will be reported, respectively.

4.4.1 Improved distributed Han's method

The improvement is expressed in the convergence characteristics. Now the variable step size α instead of a common one is used. There is a need to pre-compute and store the following parameters for the entire control scheme.

- For each $l \in L_i$: $\alpha_l = (k_\alpha)_l \cdot \alpha_0$, where k_α is the scaling vector, and α_l acts as local step size regarding l^{th} dual variable, and therefore k_α should be chosen such that the convergence rates of all s dual variables are improved.
- For each $l \in L_i$, $\bar{c}_l$ can be calculated in the following form: $\bar{c}_l = \frac{-1}{\boldsymbol{a}_l^T \cdot \boldsymbol{a}_l} \cdot \boldsymbol{H}^{-1} \cdot \boldsymbol{a}_l . \bar{c}_l$ is computed locally by a local controller with a priori knowledge of the parameter $\boldsymbol{a}_l$ and the weighting blocks on the diagonal of $\boldsymbol{H}$ that correspond to the non-zero elements $of\, \boldsymbol{a}_l$.

At the beginning of the MPC step, the current states of all subsystems are measured. The sequences of predicted states and inputs generated in the previous MPC step are shifted forward in one time interval, and then new states and new inputs of the shifted sequences are defined as zero. The new sequences are used as initial guess for solving the optimization problem in the current MPC step. This initial guess of the solution can be defined locally by each controller. At the first MPC step, we have $\boldsymbol{x}^{(0)|i} = \boldsymbol{0}$. An initial guess that is close to the optimal solution will be very helpful in Han's method that reduces the number of iterations.

Also, in the improved version of Han's method the choice of the scaling vector can significantly improve the convergence rate. As it was explained in [10], this choice should focus on improving the convergence rate of "slower convergent" dual variables. The Hessian can help to find the scaling vector. Specifically, for a subsystem i whose variables have the average weight $\overline{h}_i$ (e.g. average of entries related to i's states and inputs in the diagonal of the Hessian), and choose the scale factor $(k_\alpha)_l = \frac{1}{\overline{h}_i}$, with all $l \in L_i$. The scaling vector k_α has to be multiplied with factor $\theta \in (0,1)$ for enlarging the step sizes of all dual variables. This factor is tuned in the first MPC step. Start tuning with θ approximately equals to 1 and gradually reduce θ until it causes the algorithm to diverge (in this research the approximate value has been selected, because of the method of finding θ), then stop and choose the smallest θ such that the algorithm still converges. The choice of this factor depends on the structure of the centralized optimization problem. This factor has to be chosen only once at the first MPC step. For the next MPC step the same scaling vector can be reused. The improved distributed Han's method (IDH) is represented below.

1. **Communication to get the updated main variables**
   Each controller i communicates with its neighbor $j \in \mathcal{N}^i$ to get updated values of variables, contained in $\boldsymbol{x}^{(p-1)|i}$. Vice versa, i sends its updated variables in $\boldsymbol{x}^{(p-1)|i}$ to its neighbor as requested. After getting information from the neighbors, controller i constructs the neighborhood image $\boldsymbol{x}^{(p-1)|\mathcal{N}^i}$ using formula (4.4.2)
2. **Update intermediate variables γ_l in parallel**
   In this step, the local controllers update γ_l corresponding to each constraint l under their responsibility. Each controller i updates γ_l for each $l \in L_i$ in the equal manner as for the usual distributed Han's method (refer to section 4.4), by (4.4.8a / 4.4.8b):
   - If constraint l is an equality constraint ($1 \le l \le n_{eq}$), then
   $$\gamma_l^{(p)} = \boldsymbol{a}_l^T \cdot \boldsymbol{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l \qquad (4.4.8a)$$
   - If constraint l is an inequality constraint ($n_{eq} + 1 \le l \le s$), then
   $$\gamma_l^{(p)} = \max\left\{\boldsymbol{a}_l^T \cdot \boldsymbol{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l, 0\right\} \qquad (4.4.8b)$$
3. **Communication to get the updated intermediate variables**
   Each controller i communicates with its neighbor to get updated $\gamma_l^{(p)}$ values that the neighbors just computed in step 2.

4. **Update main variables in parallel**

   Local controller i uses all $\boldsymbol{\gamma}_l^{(p)}$ values that it has to compute an assumed neighborhood image of $\boldsymbol{x}$ using formula (4.4.9).

$$\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i} = \sum_{l \in L_{\mathcal{N}^i}} \frac{1}{\boldsymbol{\alpha}_l} \cdot \boldsymbol{\gamma}_l^{(p)} \cdot \bar{\boldsymbol{c}}_l \qquad (4.4.9)$$

   Note that $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$ has the same structure as $\boldsymbol{x}^{(p-1)|\mathcal{N}^i}$. However, it is not an exact update of the neighborhood image. Then controller i selects the value of its variables in $\boldsymbol{x}_{assumed}^{(p)|\mathcal{N}^i}$ to construct the new self image by (4.4.6).

5. **Check the termination criterion**

   In this step each local controller checks the local termination criterion. The following termination using formula (4.4.7) is proposed,ε is a threshold is.

   When all local controllers have converged, the algorithm stops and the local controller actions are implemented, otherwise the controllers proceed to step 1 to start a new iteration with $p = p + 1$.

   As a result the following main points have to be noted:

   - There is no convergence proof for the improved distributed version of Han's method;
   - The method proposed to choose scaling factors (vector) does not always work well. Sometimes after several sample steps the algorithm does not converge in the next sample step;
   - The proposed method is for quadratic programs only.

4.4.2 Modified improved distributed Han's method

There is one more modification [11] of Han's method.

The main difference between the *Improved Han's Algorithm* and the *Modified Improved Han's Algorithm* (MIDH) consists in the modified formula for the γ_l, which is used in the second step, where the update of intermediate variables takes place. The modified algorithm is shown below.

Initialize with $p = 0$. Each subsystem i uses the initial guess as $\boldsymbol{x}^{(0)|i}$. Next, for $p = 1,2,\ldots$, the following operations are executed.

1. **Communication to get the updated main variables**
   See step 1 of algorithm in Section 4.4.1.
2. **Update intermediate variables γ_l in parallel**
   Now we use the modified formulas (4.4.10a / 4.4.10b). Each controller i updates γ_l for each $l \in L_i$, (refer to section 4.4/4.4.1), except for $p = 1$. For the other iteration steps we use the same formulas as in algorithm in subsection 4.4.1:
   - If constraint l is an equality constraint ($1 \leq l \leq n_{eq}$), then refer to
     $$\gamma_l^{(1)} = \boldsymbol{a}_l^T \cdot \left(\boldsymbol{x}^{(0)|\mathcal{N}^i} - \frac{\boldsymbol{\alpha}_l}{s} \cdot \boldsymbol{H} \cdot \boldsymbol{x}^{(0)|\mathcal{N}^i} \right) - b_l \qquad (4.4.10a)$$
   - If constraint l is an inequality constraint ($n_{eq} + 1 \leq l \leq s$), then
     $$\gamma_l^{(1)} = \max\left\{ \boldsymbol{a}_l^T \cdot \left(\boldsymbol{x}^{(0)|\mathcal{N}^i} - \frac{\boldsymbol{\alpha}_l}{s} \cdot \boldsymbol{H} \cdot \boldsymbol{x}^{(0)|\mathcal{N}^i} \right) - b_l, 0 \right\} \qquad (4.4.10b)$$
3. **Communication to get the updated intermediate variables**
   See step 3 of algorithm in Section 4.4.1.
4. **Update main variables in parallel**
   See step 4 of algorithm in Section 4.4.1.
5. **Check the termination criterion**
   See step 5 of algorithm in Section 4.4.1.

   This modification leads to improvement of the convergence rate.

5. Numerical examples and simulation

5.1 Description of an aggregated water supply system

As mentioned in the introduction the comparison of the methods on the example of the water supply network will be provided, which is described in Figure 6.

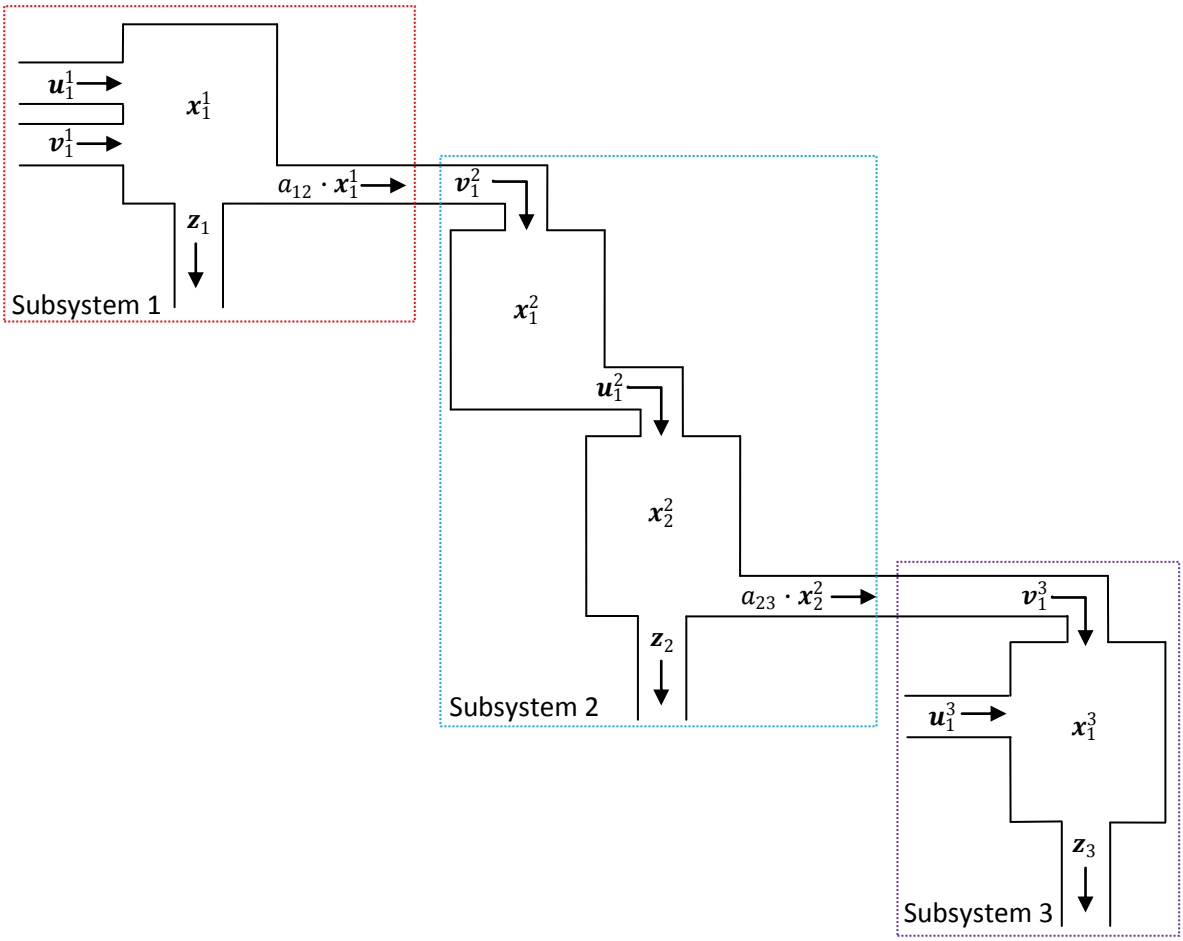

Figure 6: Aggregated water supply system

The whole system consists of three subsystems. The first subsystem is connected to the second one, and the second to the third one. The first subsystem can be associated with a village with consumption $z_{1,t} = 1800\frac{m^3}{24\ h}$, the second to the large town with $z_{2,t} = 10000\frac{m^3}{24\ h}$, the third to the small town with $z_{3,t} = 5400\frac{m^3}{24\ h}$.

The first and the third subsystem have got one water tanks, and the second one has got two water tanks. The volume of these tanks is described by the variables $x_1^1, x_1^2, x_2^2, x_1^3$ corresponding to the subsystems one, two, and three, with the upper index used as the index of the subsystem.

Table 1 shows all variables, which are important for the water supply system.

Table 1: Description of the variables for an aggregated water supply system

	Subsystem 1	Subsystem 2	Subsystem 3
State(-s)	x_1^1	x_1^2, x_2^2	x_1^3
Control	u_1^1	u_1^2	u_1^3
Pseudo-control	v_1^1	v_1^2	v_1^3
Output	$a_{12} \cdot x_1^1$	$a_{23} \cdot x_1^1$	----------
Consumption of consumers	z_1	z_2	z_3

From Table 1, the output is described as the dependence of the state. It is assumed, that this dependence can be described by a corresponding coefficient a_{12} for the connection between the first and the second subsystem and $a_{23}$for the connection between the second and the third subsystem. It is also assumed that these coefficients a_{12} and a_{23} are constant during optimization and can be calculated by the formulas (5.2.2 / 5.2.3), respectively, with the help of (5.2.1a \ 5.2.1b). The latter are the coupling equation between subsystems.

$$\boldsymbol{v}_{1,s}^2 - a_{12} \cdot \boldsymbol{x}_{1,s}^1 = 0, \qquad (5.2.1a)$$

$$\boldsymbol{v}_{1,s}^3 - a_{23} \cdot \boldsymbol{x}_{2,s}^2 = 0, \qquad (5.2.1b)$$

$$a_{12} = \frac{\boldsymbol{v}_{1,s}^2}{\boldsymbol{x}_{1,s}^1}, \qquad (5.2.2)$$

$$a_{23} = \frac{\boldsymbol{v}_{1,s}^3}{\boldsymbol{x}_{2,s}^2}, \qquad (5.2.3)$$

where $\boldsymbol{v}_{1,s}^1$, $\boldsymbol{x}_{1,s}^1$, $\boldsymbol{v}_{1,s}^3$, $\boldsymbol{x}_{2,s}^2$ represent setpoints for states and pseudo-controls. It means that these coefficients have to be calculated only once before the optimization process.

For the comparison analysis of the methods the time horizon equals to 24 hours. The variable K is defined as the number of discrete time intervals and it can be got from a predefined set {3, 6, 12, 24}. The time intervals for corresponding discrete intervals are calculated by the (5.2.4).

$$\Delta t = 24/K \tag{5.2.4}$$

Hence, $\Delta t = \{8, 4, 2, 1\}$. It means that e. g. exist three discrete intervals, and each intervals corresponds to 8 hours.

The setpoint for states $\boldsymbol{x}_{1,s}^1 = \boldsymbol{x}_{1,s}^2 = \boldsymbol{x}_{2,s}^2 = \boldsymbol{x}_{1,s}^3 = 7500\ m^3$ are defined.

For the corresponding number of intervals calculation of the setpoints for the states, controls, and pseudo-controls for all subsystems is required. The equations (5.2.5 – 5.2.9) are represented below.

$$\boldsymbol{v}_{1,s}^3 = 0.74 \cdot \frac{\sum_{k=0}^{K-1} \boldsymbol{z}_3(k)}{K} \tag{5.2.5}$$

$$\boldsymbol{u}_{1,s}^3 = 0.26 \cdot \frac{\sum_{k=0}^{K-1} \boldsymbol{z}_3(k)}{K} \tag{5.2.6}$$

$$\boldsymbol{u}_{1,s}^2 = \frac{\sum_{k=0}^{K-1} \boldsymbol{z}_2(k)}{K} + \boldsymbol{v}_{1,s}^3 \tag{5.2.7}$$

$$\boldsymbol{v}_{1,s}^2 = \boldsymbol{u}_{1,s}^2 \tag{5.2.8}$$

$$\boldsymbol{u}_{1,s}^1 = \boldsymbol{v}_{1,s}^1 = \frac{1}{2} \cdot \left(\boldsymbol{v}_{1,s}^2 + \frac{\sum_{k=0}^{K-1} \boldsymbol{z}_1(k)}{K} \right) \tag{5.2.9}$$

It is possible to describe each subsystem by the set of difference equations, the number of which is equal to the number of intervals.

For example, for $K = 3$ the following set of equation for all subsystems is given below.

Subsystem 1:

$$x_1^1(1) - (1 - a_{12}) \cdot x_1^1(0) - v_1^1(0) - u_1^1(0) = -z_1(0), \tag{5.2.10}$$

$$x_1^1(2) - (1 - a_{12}) \cdot x_1^1(1) - v_1^1(1) - u_1^1(1) = -z_1(1), \tag{5.2.11}$$

$$x_1^1(3) - (1 - a_{12}) \cdot x_1^1(2) - v_1^1(2) - u_1^1(2) = -z_1(2). \tag{5.2.12}$$

Subsystem 2:

$$x_1^2(1) - x_1^2(0) - v_1^2(0) + u_1^2(0) = 0, \tag{5.2.13}$$

$$x_2^2(1) - (1 - a_{23}) \cdot x_2^2(0) - u_1^2(0) = -z_2(0), \tag{5.2.14}$$

$$x_1^2(2) - x_1^2(1) - v_1^2(1) + u_1^2(1) = 0, \tag{5.2.15}$$

$$x_2^2(2) - (1 - a_{23}) \cdot x_2^2(1) - u_1^2(1) = -z_2(1), \tag{5.2.16}$$

$$x_1^2(3) - x_1^2(2) - v_1^2(2) + u_1^2(2) = 0, \tag{5.2.17}$$

$$x_2^2(3) - (1 - a_{23}) \cdot x_2^2(2) - u_1^2(2) = -z_2(2), \tag{5.2.18}$$

Subsystem 3:

$$x_1^3(1) - x_1^3(0) - v_1^3(0) - u_1^3(0) = -z_3(0), \tag{5.2.19}$$

$$x_1^3(2) - x_1^3(1) - v_1^3(1) - u_1^3(1) = -z_3(1), \tag{5.2.20}$$

$$x_1^3(3) - x_1^3(2) - v_1^3(2) - u_1^3(2) = -z_3(2), \tag{5.2.21}$$

The cost function for the subsystem is described by (5.2.22). There is no estimation of the state, control or pseudo-control variable, but the quadratic error between actual value of these variables and the corresponding setpoints.

$$J_i = \frac{1}{2} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{s,i})^T \cdot \boldsymbol{Q}_i \cdot \left(\boldsymbol{x}_i - \boldsymbol{x}_{s,i}\right) + \frac{1}{2} \cdot (\boldsymbol{u}_i - \boldsymbol{u}_{s,i})^T \cdot \boldsymbol{R}_i \cdot \left(\boldsymbol{u}_i - \boldsymbol{u}_{s,i}\right) + \frac{1}{2} \cdot (\boldsymbol{v}_i - \boldsymbol{v}_{s,i})^T \cdot \boldsymbol{S}_i \cdot \left(\boldsymbol{v}_i - \boldsymbol{v}_{s,i}\right), \tag{5.2.22}$$

with $\boldsymbol{x}_i$ corresponds to total state vector of subsystem, and $\boldsymbol{x}_{s,i}$ corresponds to total state setpoint vector, and $\boldsymbol{u}_i$, $\boldsymbol{u}_{s,i}$, $\boldsymbol{v}_i$, $\boldsymbol{v}_{s,i}$ analogously; $\boldsymbol{Q}_i, \boldsymbol{R}_i, \boldsymbol{S}_i$ are quadratic positive-definite diagonal weighting matrices. Their dimension corresponds to the number of rows of variables to which they belong.

With respect to the cost function the equations (5.2.10) – (5.2.21), corresponding to the difference between actual variables and their setpoints, have to be written.

It is provided in matrix-vector form, because it is the most comfortable form for the further implementation in MATLAB®[1].

Subsystem 1:

$$[\boldsymbol{A}_1, \boldsymbol{B}_1, \boldsymbol{C}_1] \cdot \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{u}_1 \\ \boldsymbol{v}_1 \end{bmatrix} = \boldsymbol{Z}_1 \cdot \boldsymbol{z}_1, \tag{5.2.23}$$

[1]©2012 The MathWorks, Inc.MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

with $\boldsymbol{x}_1 = \tilde{\boldsymbol{x}}_1 + \boldsymbol{x}_{s,1}, \boldsymbol{u}_1 = \tilde{\boldsymbol{u}}_1 + \boldsymbol{u}_{s,1}, \boldsymbol{v}_1 = \tilde{\boldsymbol{v}}_1 + \boldsymbol{v}_{s,1}$

$$[\boldsymbol{A}_1, \boldsymbol{B}_1, \boldsymbol{C}_1] \cdot \begin{bmatrix} \tilde{\boldsymbol{x}}_1 \\ \tilde{\boldsymbol{u}}_1 \\ \tilde{\boldsymbol{v}}_1 \end{bmatrix} = \boldsymbol{Z}_1 \cdot \boldsymbol{z}_1 - [\boldsymbol{A}_1, \boldsymbol{B}_1, \boldsymbol{C}_1] \cdot \begin{bmatrix} \boldsymbol{x}_{s,1} \\ \boldsymbol{u}_{s,1} \\ \boldsymbol{v}_{s,1} \end{bmatrix}. \qquad (5.2.24)$$

Subsystem 2:

$$[\boldsymbol{A}_2, \boldsymbol{B}_2, \boldsymbol{C}_2] \cdot \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{u}_2 \\ \boldsymbol{v}_2 \end{bmatrix} = \boldsymbol{Z}_2 \cdot \boldsymbol{z}_2, \qquad (5.2.25)$$

with $\boldsymbol{x}_2 = \tilde{\boldsymbol{x}}_2 + \boldsymbol{x}_{s,2}, \boldsymbol{u}_2 = \tilde{\boldsymbol{u}}_2 + \boldsymbol{u}_{s,2}, \boldsymbol{v}_2 = \tilde{\boldsymbol{v}}_2 + \boldsymbol{v}_{s,2}$

$$[\boldsymbol{A}_2, \boldsymbol{B}_2, \boldsymbol{C}_2] \cdot \begin{bmatrix} \tilde{\boldsymbol{x}}_2 \\ \tilde{\boldsymbol{u}}_2 \\ \tilde{\boldsymbol{v}}_2 \end{bmatrix} = \boldsymbol{Z}_2 \cdot \boldsymbol{z}_2 - [\boldsymbol{A}_2, \boldsymbol{B}_2, \boldsymbol{C}_2] \cdot \begin{bmatrix} \boldsymbol{x}_{s,2} \\ \boldsymbol{u}_{s,2} \\ \boldsymbol{v}_{s,2} \end{bmatrix}. \qquad (5.2.26)$$

Subsystem 3:

$$[\boldsymbol{A}_3, \boldsymbol{B}_3, \boldsymbol{C}_3] \cdot \begin{bmatrix} \boldsymbol{x}_3 \\ \boldsymbol{u}_3 \\ \boldsymbol{v}_3 \end{bmatrix} = \boldsymbol{Z}_3 \cdot \boldsymbol{z}_3, \qquad (5.2.27)$$

with $\boldsymbol{x}_3 = \tilde{\boldsymbol{x}}_3 + \boldsymbol{x}_{s,3}, \boldsymbol{u}_3 = \tilde{\boldsymbol{u}}_3 + \boldsymbol{u}_{s,3}, \boldsymbol{v}_3 = \tilde{\boldsymbol{v}}_3 + \boldsymbol{v}_{s,3}$

$$[\boldsymbol{A}_3, \boldsymbol{B}_3, \boldsymbol{C}_3] \cdot \begin{bmatrix} \tilde{\boldsymbol{x}}_3 \\ \tilde{\boldsymbol{u}}_3 \\ \tilde{\boldsymbol{v}}_3 \end{bmatrix} = \boldsymbol{Z}_3 \cdot \boldsymbol{z}_3 - [\boldsymbol{A}_3, \boldsymbol{B}_3, \boldsymbol{C}_3] \cdot \begin{bmatrix} \boldsymbol{x}_{s,3} \\ \boldsymbol{u}_{s,3} \\ \boldsymbol{v}_{s,3} \end{bmatrix}. \qquad (5.2.28)$$

In equations (5.2.23) – (5.2.28) $\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{C}_i, \boldsymbol{Z}_i$ are used as dynamic, control, pseudo-control, and disturbance matrices, respectively. $\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{u}}_i, \tilde{\boldsymbol{v}}_i$ with $i = 1..3$ are the difference between state, control, and pseudo-control vectors and their setpoint vectors.

As it was presented a different numbers of discrete intervals were used. That means that the water demand in each discrete interval has to be defined. For this goal we use the statistical data. The water demand is shown in Figure 6 for the each hour per day, and presented in percent. The total daily water demand is known, and it can be calculated in cubic meters.

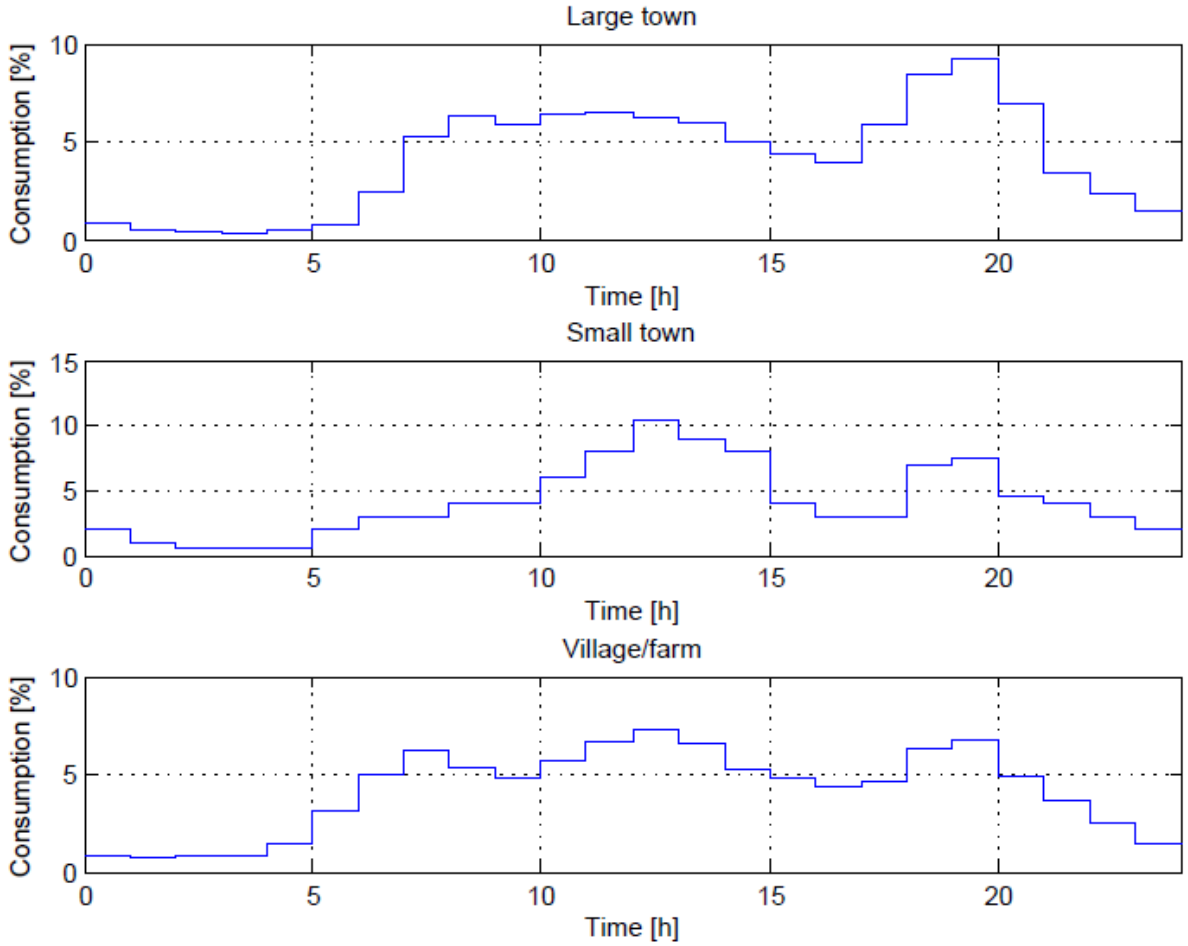

Figure 6: Daily water consumption

In order to have a properly working water supply system without failures the boundaries of acceptable values for states, controls, and pseudo-controls have to be entered. The following lower and upper boundaries are assumed (5.2.29 – 5.2.37). These factors 0.5 and 0.75 (see below) were selected based on the assumption that the water level in the water tanks can varywithin the limits. Coefficient 0.75 was chosen different from the other because of the second system is a big city, and consequently, the level of water in the tank and water demand may have a higher amplitude of fluctuation.

$$5000 - \boldsymbol{x}_{s,1} \leq \tilde{\boldsymbol{x}}_1 \leq 10000 - \boldsymbol{x}_{s,1}, \quad (5.2.29)$$

$$\left(\boldsymbol{u}_{s,1} - 0.5 \cdot \boldsymbol{u}_{s,1}\right) - \boldsymbol{u}_{s,1} \leq \tilde{\boldsymbol{u}}_1 \leq \left(\boldsymbol{u}_{s,1} + 0.5 \cdot \boldsymbol{u}_{s,1}\right) - \boldsymbol{u}_{s,1}, \quad (5.2.30)$$

$$\left(\boldsymbol{v}_{s,1} - 0.5 \cdot \boldsymbol{v}_{s,1}\right) - \boldsymbol{v}_{s,1} \leq \tilde{\boldsymbol{v}}_1 \leq \left(\boldsymbol{v}_{s,1} + 0.5 \cdot \boldsymbol{v}_{s,1}\right) - \boldsymbol{v}_{s,1}, \quad (5.2.31)$$

$$2000 - \boldsymbol{x}_{s,2} \leq \tilde{\boldsymbol{x}}_2 \leq 10000 - \boldsymbol{x}_{s,2}, \quad (5.2.32)$$

$$\left(\boldsymbol{u}_{s,2} - 0.75 \cdot \boldsymbol{u}_{s,2}\right) - \boldsymbol{u}_{s,2} \leq \tilde{\boldsymbol{u}}_2 \leq \left(\boldsymbol{u}_{s,2} + 0.75 \cdot \boldsymbol{u}_{s,2}\right) - \boldsymbol{u}_{s,2}, \quad (5.2.33)$$

$$\left(\boldsymbol{v}_{s,2} - 0.75 \cdot \boldsymbol{v}_{s,2}\right) - \boldsymbol{v}_{s,2} \leq \widetilde{\boldsymbol{v}}_2 \leq \left(\boldsymbol{v}_{s,2} + 0.75 \cdot \boldsymbol{v}_{s,2}\right) - \boldsymbol{v}_{s,2}, \quad (5.2.34)$$

$$5000 - \boldsymbol{x}_{s,3} \leq \widetilde{\boldsymbol{x}}_3 \leq 10000 - \boldsymbol{x}_{s,3}, \quad (5.2.35)$$

$$\left(\boldsymbol{u}_{s,3} - 0.5 \cdot \boldsymbol{u}_{s,3}\right) - \boldsymbol{u}_{s,3} \leq \widetilde{\boldsymbol{u}}_3 \leq \left(\boldsymbol{u}_{s,3} + 0.5 \cdot \boldsymbol{u}_{s,3}\right) - \boldsymbol{u}_{s,3}, \quad (5.2.36)$$

$$\left(\boldsymbol{v}_{s,3} - 0.5 \cdot \boldsymbol{v}_{s,3}\right) - \boldsymbol{v}_{s,3} \leq \widetilde{\boldsymbol{v}}_1 \leq \left(\boldsymbol{v}_{s,3} + 0.5 \cdot \boldsymbol{v}_{s,3}\right) - \boldsymbol{v}_{s,3}. \quad (5.2.37)$$

In addition to the lower and upper limits, the values of all variables have to be evaluated. It can be provided by using the partial objective function for each subsystem using (5.2.22), and more specifically by weight matrices $\boldsymbol{Q}_i, \boldsymbol{R}_i, \boldsymbol{S}_i$. We propose the following weights for them:

$$\boldsymbol{Q}_i = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \boldsymbol{R}_i = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \boldsymbol{S}_i = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 20 \end{bmatrix}. \quad (5.2.38)$$

The high amplitude fluctuations in the water storage and the low amplitude in the connected pipes are admitted. Certainly the values of weighting matrices can be increased by an order estimation of values, but it does not make much sense since it will become more difficult to find the minimum of the objective function, hence the right values of states, controls, and pseudo-controls. This is due to the fact that the objective function changes its shape (become more stretched) so that it is difficult to find a minimum.

For simulations, the initial and final values of states, in other words the volume in water tanks at the beginning and end of the time horizon, have to be equal to their setpoints, which means the difference between actual state and setpoint equals to zero. All other state values are optimization variables.

Since the water demand is stochastically changing in different time intervals the proposed water supply system has to be analyzed with disturbances in the consumption. The water consumption depends on the different factors such as air temperature, season, etc. It is assumed that the maximal deviation from non-disturbed condition equals to ± 10 %. The disturbance forms in a following approach. The consumptions are calculated for each discrete time interval based on the dependence, which is described in Figure 6. Then this value is assumed as the mean value in this interval. After that the Normal Distribution Law is applied and only the values in ± 10 % range around mean value had been selected.

In next subsections (5.2 / 5.3) the results of the simulations will be presented. All experiments have been provided using CPU Intel® Core™ Duo T5750, $f_{CPU} = 1.4$ GHz.

5.2 Results of the hierarchical case simulation

In case of the hierarchical control structure The *Interaction Balance Method* was used, which is described in subsection 3.3. Within the termination criterion of this method, the threshold in the amount of $\mathcal{E} = 10$ on the coordinator level has applied, thereby

$$\|\nabla_{\boldsymbol{\lambda}}\Phi(\boldsymbol{\lambda})\|_2 < \mathcal{E}. \tag{5.2.39}$$

The use of such a large threshold based on the fact that the lesser the value of threshold then larger the time of calculation the values of the state, control, and pseudo-control vectors, but their accuracy does not practically change. Referred to the equation (3.3.10) the step size α has to be chosen. It is proposed $\alpha = 25$. As it was described in subsection 3.3, a common step size or an iteratively changing can be used. For this research the latter is used and the simulation is provided with different decrement such as $dec = \{0.05, 0.03, 0.01\}$. Reducing the value of the decrement increases the number of iterations in the coordinator under the condition $K = 3$, hence increasing the time of calculation of values (refer to Tables 2 – 4). But for $K = 6$ or $K = 12$ reducing the decrement leads to the opposite effect: the time of calculation is decreased.

In the following tables the simulation results of one day prediction for different $K = 3, 6, 12, 24$ are presented. The results consist of the number of iteration on the coordinator level, CPU time (coordinator time and optimizer time), and of the number of iteration in each subsystem. For an actual condition of the state, control, and pseudo-control vector for i.e. offline solutions (Table 2, gray marked line) for all subsystems in the hierarchical case simulation refer to the (Fig. 7a – Fig. 7c).

Table 2: Optimization performance data for $K = 3, dec = 0.05, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
14	0	7,0512452	155	177	160
11	0,0312002	5,6628363	130	134	155
10	0	5,2416336	168	137	136
10	0	5,4756351	132	208	120
45	**0,0312002**	**23,4313502**	**585**	**656**	**571**

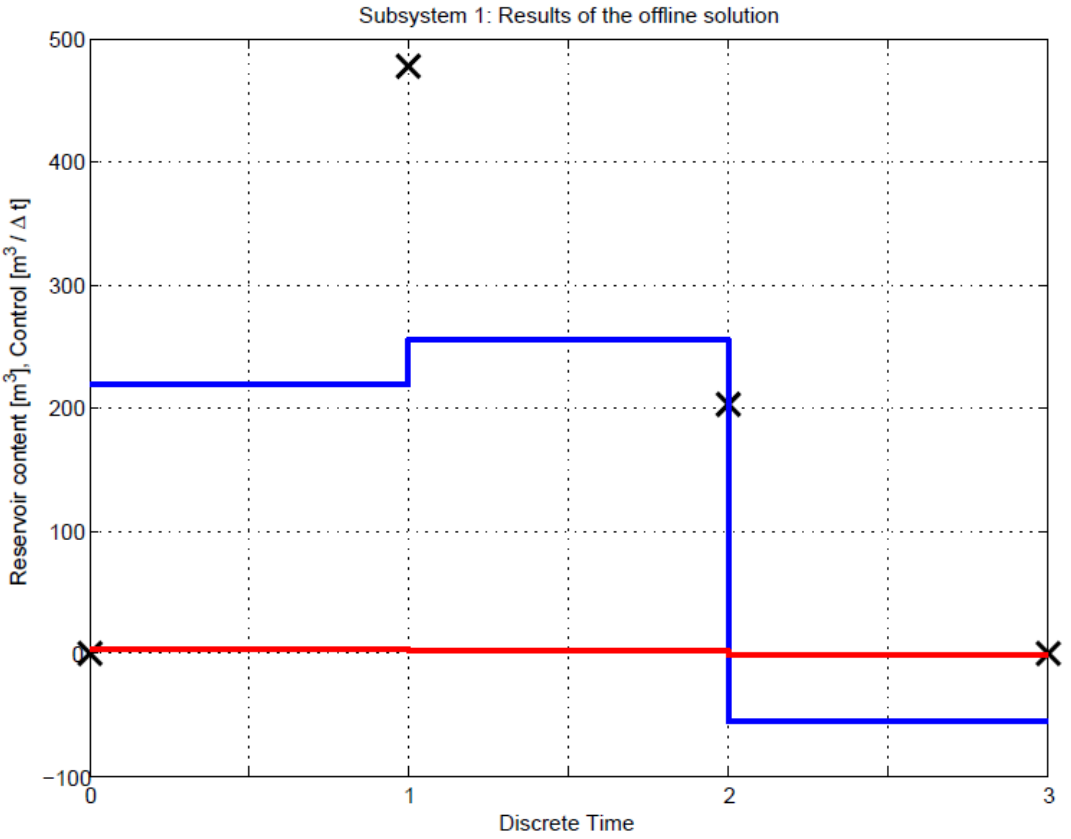

Figure 7a: Results of the offline solution for subsystem 1

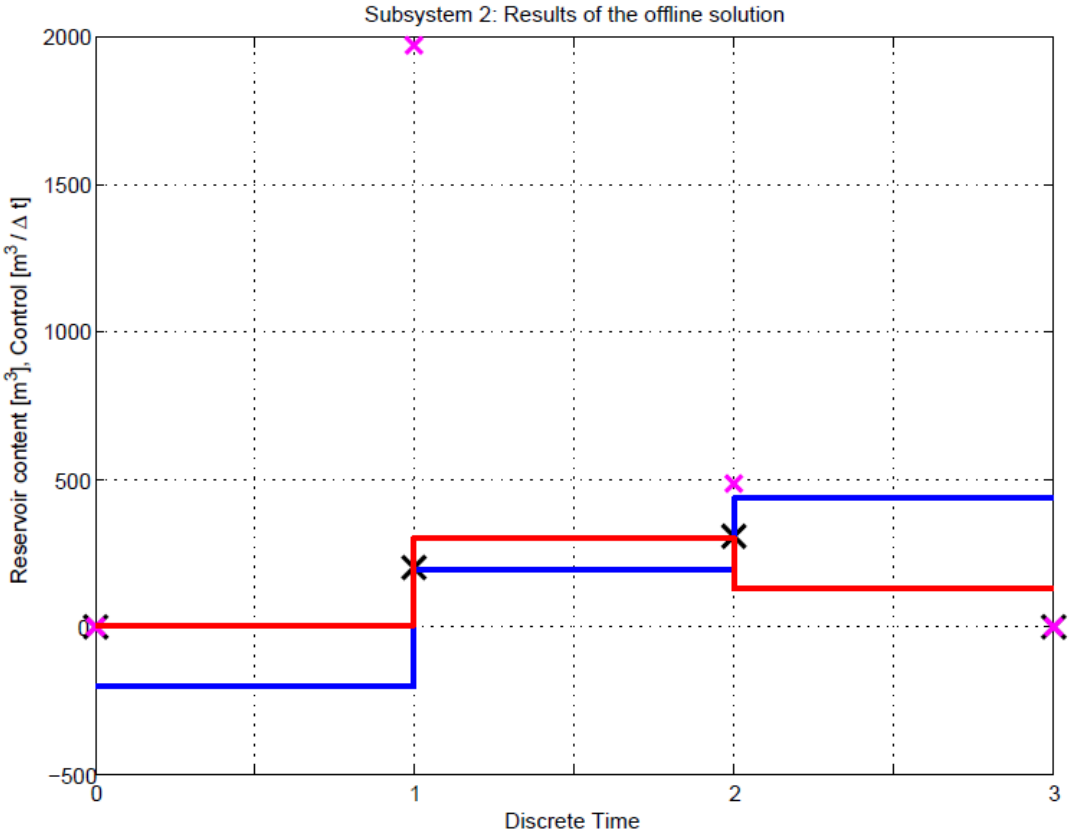

Figure 7b: Results of the offline solution for subsystem 2

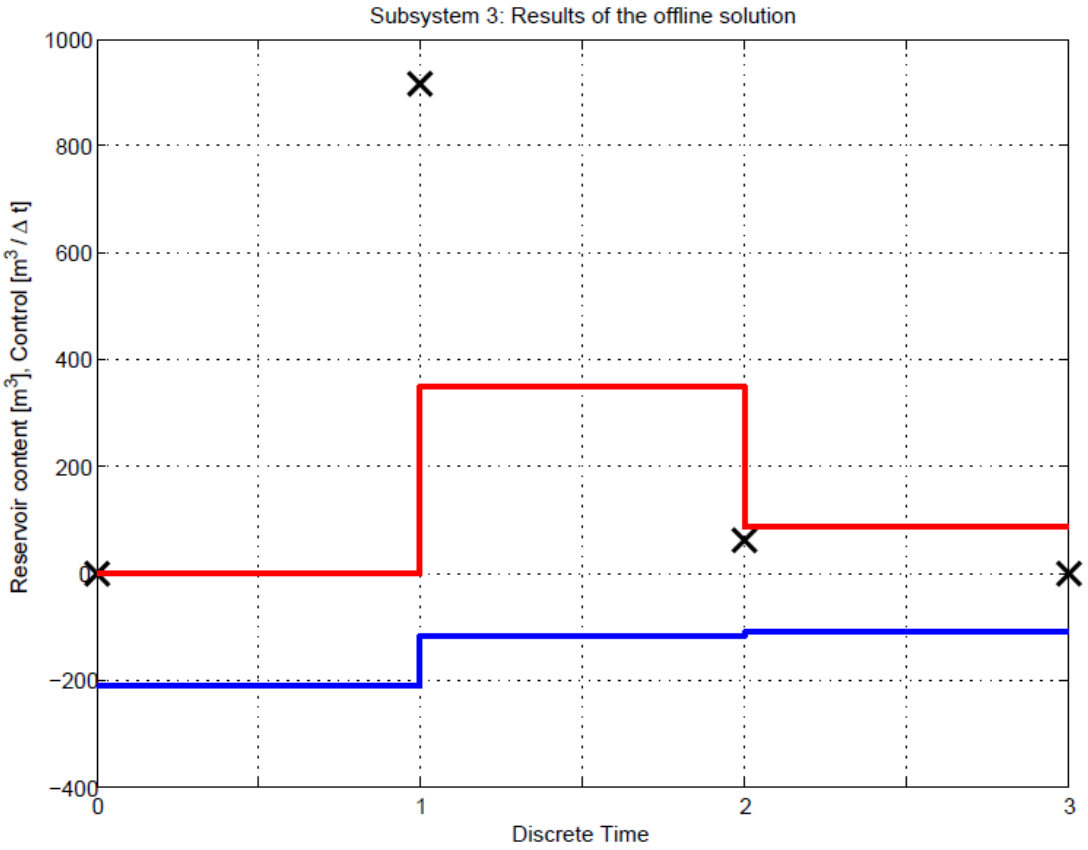

Figure 7c: Results of the offline solution for subsystem 3

Comments for Fig. 7a and Fig. 7c: black marked – system states, blue line – control variable, red line – pseudo-control variable.

Comment for Fir. 7b: black marked – the first system state, magenta marked – second system state, blue and red line are the same as for Fig. 7a and Fig. 7c

Table 3: Optimization performance data for $K = 3, dec = 0.03, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
13	0	7,0668453	143	207	150
14	0	5,6472362	180	171	188
12	0	5,304034	185	142	174
12	0,0312002	5,1636331	158	193	143
51	**0,0312002**	**23,1817486**	**666**	**713**	**655**

Table 4: Optimization performance data for $K = 3, dec = 0.01, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
23	0	9,9528638	260	302	239
26	0	10,2492657	299	311	347
23	0,0312002	9,9372637	366	270	284
23	0	10,3740665	286	319	259
95	**0,0312002**	**40,5134597**	**1211**	**1202**	**1129**

Table 5: Optimization performance data for $K = 6,\ dec = 0.05, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
14	0	15,8809018	297	375	451
6	0	13,8060885	169	213	204
7	0	13,2912852	178	249	204
13	0	12,8700825	414	330	320
17	0	16,5205059	330	659	442
15	0	14,4768928	294	407	361
6	0	16,2397041	142	324	154
78	**0**	**103,0854608**	**1824**	**2557**	**2136**

Table 6: Optimization performance data for $K = 6, dec = 0.03, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
12	0	14,9136956	246	341	379
6	0	11,9652767	146	205	180
7	0	12,5736806	213	278	208
11	0	11,0136706	313	281	310
15	0	14,5860935	311	579	377
13	0,0312002	12,9168828	246	376	298
6	0	14,8356951	150	318	159
70	**0,0312002**	**92,8049949**	**1625**	**2378**	**1911**

Table 7: Optimization performance data for $K = 6, dec = 0.01, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
11	0	13,416086	233	289	342
6	0	11,0292707	172	204	182
7	0	12,012077	195	334	214
10	0	9,7656626	278	244	261
13	0	13,2288848	258	553	355
12	0,0312002	11,2008718	231	332	279
6	0,0312002	12,168078	147	212	171
65	**0,0624004**	**82,8209309**	**1514**	**2168**	**1804**

Table8: Optimization performance data for $K = 12, dec = 0.05, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
20	0	133,6616568	1214	1227	980
10	0	142,8345156	575	871	560
30	0	176,0783287	1182	1634	1730
13	0	192,973237	702	1088	791
33	0	223,3466317	1158	2005	2902
21	0	230,3978769	727	1567	1859
6	0	223,6742338	296	507	364
5	0	222,1766242	231	376	320
9	0	215,6713825	556	568	380
27	0	206,1085212	1748	1972	882
16	0	209,1661408	977	1193	612
34	0	196,8108616	1211	2335	1510
24	0	209,4781428	1012	1568	1276
248	**0**	**2582,378154**	**11589**	**16911**	**14166**

Table 9: Optimization performance data for $K = 12, dec = 0.03, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
16	0	116,3923461	973	1043	762
9	0	120,9631754	500	903	510
21	0	128,1860217	839	1171	1212
11	0	148,6533529	601	984	731
23	0	151,5705716	827	1362	1759
17	0	156,5314034	590	1270	1263
6	0	151,1961692	304	477	326
5	0	153,5673844	235	376	317
8	0,0312002	144,0357233	429	541	331
20	0	139,7456958	1279	1490	640
14	0	149,9013609	862	1093	526
23	0	139,7144956	864	1527	1037
19	0	147,8733479	827	1192	1066
192	**0,0312002**	**1848,331048**	**9130**	**13429**	**10480**

Table 10: Optimization performance data for $K = 12, dec = 0.01, \alpha = 25$

Iterations in Coordinator	Time in coordinator [s]	Time in Optimizer [s]	Iterations in subsystem 1	Iterations in subsystem 2	Iterations in subsystem 3
14	0	102,1962551	772	933	714
9	0	108,2178937	493	836	537
18	0,0312002	112,4611209	708	976	1078
10	0,0312002	130,5884371	509	899	733
19	0	136,8596773	691	1072	1830
15	0	144,1137238	525	1104	1448
5	0	144,1761242	256	447	356
5	0	138,3884871	235	361	287
8	0	134,4260617	467	574	317
17	0	120,0895698	1102	1193	567
12	0	118,1083571	738	822	477
19	0	121,524779	692	1316	812
16	0	135,7832704	693	1041	789
167	**0,0624004**	**1646,933757**	**7881**	**11574**	**9945**

Comments to the tables (Table 2 – Table 10): the first line (gray marked) is the so-called offline solution without disturbances. Other rows explain the one interval prediction (calculation horizon is shifted forward by one interval), except the last one (bold marked), which means the sum of all superior lines.

With growing K the time of optimization grows nonlinear. We have provided the complete optimization process (offline solution with one day prediction) with $K = 24$, but the CPU time equals approximately 3,5 hours.

5.3 Results of the distributed case simulation

In this subsection the simulation result of the *Distributed Han's algorithm* and its improved and modified improved version will be presented.

As for the *Interaction Balance Method* the termination criterion has to be chosen. The estimation is based on the Euclidian norm of the difference between the results of previous and following iteration (5.3.1). When the norm is lower or equal to the threshold $\mathcal{E} = 0.1$, then the optimization process terminates and the values of state, control, and pseudo-control vectors for each subsystem become known.

$$\left\| \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \\ \boldsymbol{v}_i \end{bmatrix}^{q+1} - \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \\ \boldsymbol{v}_i \end{bmatrix}^{q} \right\|_2 \leq \mathcal{E}, \qquad (5.3.1)$$

where q is an iteration counter, i is the subsystem index.

The results of distributed, improved distributed, and modified improved distributed Han's algorithm for the next one day prediction are presented below. For Table 11 – Table 22: gray marked – offline solution, bold marked – the sum of all superior lines.

Table 11: DH: Optimization performance data for $K = 3$

Iterations	Time [s]
1477	13,8996891
381	3,4944224
1407	12,3396791
308	2,496016
3573	**32,2298066**

Table 12: DH: Optimization performance data for $K = 6$

Iterations	Time [s]
1330	15,5688998
1196	12,0588773
186	1,8408118
727	6,8484439
1753	16,4737056
668	6,3336406
1494	14,04009
7354	**73,164469**

Table 13: DH: Optimization performance data for $K = 12$

Iterations	Time [s]
1596	20,6389323
958	9,9996641
1671	16,3489048
868	8,4552542
1237	12,324079
359	3,5724229
1968	19,8433272
500	5,0856326
532	5,4132347
973	10,140065
14281	**148,7001532**

Table 14: DH: Optimization performance data for $K = 24$

Iterations	Time [s]
3790	120,2143706
2119	67,2052308
1065	33,3998141
2162	68,1256367
1409	45,1466894
1092	34,476221
955	30,0457926
979	31,0285989
820	25,9585664
619	19,6093257
1246	39,780255
668	21,2005359
1726	55,692357
628	20,3581305
1245	41,3870653
1013	33,0878121
1290	43,1810768
831	27,3625754
1841	62,712402
993	34,3670203
1107	38,1578446
1290	45,9422945
927	32,9942115
2334	82,4621286
2765	96,876621
34914	**1150,772577**

Table 15: IDH: Optimization performance data for $K = 3$

Iterations	Time [s]
1013	5,3664344
220	1,092007
990	5,1012327
185	0,8580055
2408	**12,4176796**

Table 16: IDH: Optimization performance data for $K = 6$

Iterations	Time [s]
1131	8,3304534
992	7,3164469
236	1,7472112
909	6,3492407
1237	8,5488548
473	3,3540215
1037	7,4256476
6015	**43,0718761**

Table 17: IDH: Optimization performance data for $K = 12$

Iterations	Time [s]
1242	13,6344874
816	8,6580555
2320	24,1645549
2208	22,9009468
2605	27,2221745
942	9,828063
1406	14,8824954
1122	11,7936756
584	6,396041
663	7,4412477
707	7,956051
1531	17,5189123
839	9,0324579
16985	**181,429163**

Table 18: IDH: Optimization performance data for $K = 24$

Iterations	Time [s]
1935	57,7047699
2536	74,9428804
985	29,2345874
1936	41,1374637
2318	50,4507234
1934	42,0266694
2955	64,2880121
1375	29,8273912
2457	53,6019436
574	12,3552792
1421	31,3406009
2177	48,0951083
2471	55,5051558
2260	50,9967269
1796	40,9814627
1259	28,4077821
1242	28,1581805
567	12,8544824
1170	26,6449708
1332	30,0925929
3944	89,2637722
632	14,3832922
3296	74,9896807
2193	50,5599241
5046	116,2987455
49811	**1154,142198**

Table 19: MIDH: Optimization performance data for $K = 3$

Iterations	Time [s]
1013	5,4444349
963	4,8672312
199	1,0296066
230	1,0764069
2405	**12,4176796**

Table 20: MIDH: Optimization performance data for $K = 6$

Iterations	Time [s]
1131	8,5956551
240	1,6536106
221	1,5756101
571	4,3212277
1223	8,580055
1384	10,1244649
568	6,2712402
5338	**41,1218636**

Table 21: MIDH: Optimization performance data for $K = 12$

Iterations	Time [s]
1242	14,196091
707	7,8936506
1544	17,316111
1038	11,4816736
1555	17,3005109
1313	14,6484939
797	8,9544574
1555	17,3785114
988	11,1540715
742	8,3616536
809	8,9856576
1587	18,0025154
1346	15,2724979
15223	**170,9458958**

Table 22: MIDH: Optimization performance data for $K = 24$

Iterations	Time [s]
1935	40,716261
3244	67,0648299
2040	42,2450708
4244	88,1561651
1575	32,7446099
2351	48,8439131
2995	62,6344015
1594	33,5558151
1750	36,7850358
3838	82,3997282
2530	54,7875512
938	20,4673312
1809	39,4214527
1938	41,9018686
3396	73,9600741
3374	74,0224745
1651	36,4418336
3251	71,7916602
3969	88,1093648
1077	24,1645549
2202	50,4039231
4255	97,0638222
1784	41,2154642
2042	47,1903025
2640	61,7607959
62422	**1357,848304**

6. Comparison of the results and conclusions

In this section comparison of the simulation results and conclusion will be introduced.

The following prerequisites are valid for all experiments:

1. All experiments have been provided with the same CPU frequency $f_{CPU} = 1.4$ GHz
2. Technical characteristics of computer:
   - ❖ Windows Vista Business Service Pack 2, 32-bit
   - ❖ Intel® Core™ 2 Duo T5750
   - ❖ Memory (RAM) 2 GB
   - ❖ MATLAB / Simulink 7.9.0.529 (R2009b)
3. For the prediction the same consumption (disturbance) vector $\boldsymbol{z}_i$, for $i = 1, \dots, 3$ was used.
4. Initial condition or in other words deviation from setpoints of the solution for all state, control, and pseudo-control vectors is equal to zero.

Under the optimization there were no constraint violations (5.2.29 – 5.2.37) and the equality constraints are satisfied (5.2.10 – 5.2.21).

In the Table 23 the comparison of the optimization results is provided and the optimization time for the different number of intervals K is considered. Only offline solution will be analyzed.

Table 23: Optimization time demand for the offline solution

	$K = 3$	$K = 6$	$K = 12$	$K = 24$
Hierarchical	7.05 [s]	13.41 [s]	102.20 [s]	~667 [s]
DH	13.89 [s]	15.57 [s]	20.63 [s]	120.21 [s]
IDH	5.36 [s]	8.33 [s]	13.63 [s]	57.7 [s]
MIDH	5.44 [s]	8.59 [s]	14. 19 [s]	40.71 [s]

For a small number of intervals $K = 3$ or $K = 6$ it is proposed to use hierarchical structure despite the fact that for IDH and MIDH the optimization requires the time, which is smaller than for hierarchical case, but the tolerance of the state, control, and pseudo-control vectors have the more accurate result. It means that using hierarchical method for small number of intervals the values of all vectors most closely reflect the more

realistic picture of the process. For a large number of intervals it is recommended to apply distributed structure.

The results of vectors (state, control, and pseudo-control) for all used methods for case $K = 3$ are presented below (Table 24). All values performed in cubic meters.

Table 24: Optimization performance data for the offline solution

Total vector				
	IBM	DH	IDH	MIDH
$x_1^1(0)$	8,546e-21	1,104	0,577	0,576
$x_1^1(1)$	485,909	436,468	446,171	446,171
$x_1^1(2)$	206,523	220,298	230,0921	230,092
$x_1^1(3)$	9,852e-21	0,196	0,104	0,105
$u_1^1(0)$	231,526	89,583	94,928	94,928
$u_1^1(1)$	261,561	144,737	148,042	148,042
$u_1^1(2)$	-59,387	-31,156	-33,205	-33,205
$v_1^1(0)$	-0,895	89,583	94,928	94,928
$v_1^1(1)$	-3,550	144,737	148,042	148,042
$v_1^1(2)$	1,469	-31,156	-33,205	-33,205
$x_1^2(0)$	2,264e-20	1,370	0,934	0,934
$x_2^2(0)$	4,649e-20	1,039	0,713	0,713
$x_1^2(1)$	200,886	183,745	181,525	181,525
$x_2^2(1)$	1970,967	2003,326	2000,728	2000,728
$x_1^2(2)$	305,187	263,220	263,645	263,645
$x_2^2(2)$	482,493	513,305	511,085	511,085
$x_1^2(3)$	-9,189e-21	-1,355	-0,926	-0,926
$x_2^2(3)$	7,669e-22	-1,749	-1,204	-1,204
$u_1^2(0)$	-202,365	-174,206	-175,480	-175,480
$u_1^2(1)$	198,236	198,503	199,652	199,653
$u_1^2(2)$	439,864	408,109	411,952	411,952
$v_1^2(0)$	-1,479	2,705	1,380	1,381
$v_1^2(1)$	302,536	272,596	278,098	278,098
$v_1^2(2)$	134,676	138,130	143,687	143,687
$x_1^3(0)$	6,044e-20	0,004	0,031	0,030
$x_1^3(1)$	916,602	913,264	913,483	913,483
$x_1^3(2)$	60,928	59,434	59,544	59,544
$x_1^3(3)$	2,537e-20	-0,008	-0,032	-0,032
$u_1^3(0)$	-208,402	-212,161	-211,749	-211,749
$u_1^3(1)$	-116,742	-120,835	-120,401	-120,401
$u_1^3(2)$	-110,649	-114,891	-114,446	-114,446
$v_1^3(0)$	0,004	0,228	0,149	0,149
$v_1^3(1)$	350,068	355,878	355,374	355,375
$v_1^3(2)$	85,720	72,772	90,811	90,810

This table confirms that for a small number of intervals a hierarchical structure for optimization process has to be used rather than distributed because the start and end value of the states have to be equal zero. For the results of other vectors refer to MATLAB-files on the CD.

7. Summary

In this thesis the application of optimization techniques was presented for hierarchical (The *IBM*) and distributed (*Han's algorithm*) methods. The goal is to recognize which control technique fits best for controlling the water supply or other similar systems and to provide the comparison analysis of the methods.

Simulations were conducted using the software MATLAB®. To solve the task the following approach was chosen. The search of the literature regarding the hierarchical and distributed methods has been provided and MPC technology has been studied. Debugging has been done on simplified water supply system and then these methods have been supplemented with the relevant description of the aggregated water supply system, which better reflects the processes occurring in the real situation.

Based on the research, the following inferences can be made. For optimization of systems with a small number of variables and couplings between them should be used a hierarchical structure, as these methods allow to achieve more accurate results in a shorter period of time. For systems with a large number of variables and couplings the distributed structure for optimization should be used. The number of variables and couplings, CPU frequency and usage affect the optimization process; hence the different CPU time for optimization is required. It should be noted that MATLAB® is not the fastest interpreter. Distributed Han's algorithm, its improved, and modified versions are not yet fully investigated, especially with regard to convergence and parameter θ, meaning the way of its determination. In this paper for determination of this parameter *Bisection Interval* method was used. The disturbance (consumption) vector influences the optimization process, which means that under the different values of consumption the distributed algorithms can solve the same optimization problems with different CPU time demand.

The author proposes for further work:

- Use another programming language such as C++.
- Use a different method to find θ.
- Use a faster and more powerful computer.
- Use the Parallel Computing Toolbox™ in MATLAB®.

Bibliography

1. D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. Scokaert: Constrained Model Predictive Control: Stability and Optimality, Survey Paper, Automatica 36, 2000, pp. 789 – 814.
2. S. Hopfgarten: Hierarchische Steuerungssysteme, Lectures, TU Ilmenau, 2010.
3. M. D. Doan, T. Keviczky, B. Schutter: A Dual Decomposition-Based Optimization Method with Guaranteed Primal Feasibility for Hierarchical MPC Problems, Preprints of the 18th IFAC World Congress Milano (Italy) August 28 – September 2, 2011, pp. 392 – 397.
4. B. T. Stewart, J. B. Rawlings, and S. J. Wright: Hierarchical Cooperative Distributed Model Predictive Control, In Proceedings of the American Control Conference, Baltimore, Maryland, June 2010.
5. G. Anandalingam and T. L. Friesz: Hierarchical optimization: An Introduction, Annals of Operations Research 34, 1992, pp. 1 – 11.
6. R. Scattolini: Architectures for Distributed and Hierarchical Model Predictive Control – A Review, Journal of Process Control 19, 2009, pp. 723 – 731.
7. R. Haftka and Z. Gürdal: Elements of Structural Optimization, Third Revised and Expanded Edition, Kluwer Academic Publishers, 1992
8. S.-P. Han and G. Lou: A Parallel Algorithm for a Class of Convex Programs. Siam J. Control and Optimization, Vol. 26, No. 2 March 1988, pp. 345 – 355.
9. M. D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter: A Distributed Version of Han's Method for DMPC of Dynamically Coupled Systems with Coupled Constraints, *Proceeding of the 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys 2009)*, Venice, Italy, September 2009, pp. 240 – 245.
10. M. D. Doan, T. Keviczky, B. De Schutter: An Improved Distributed Version of Han's Method for distributed MPC of Canal Systems, *Proceedings of the 12th IFAC Symposium on Large Scale Systems, Theory and Applications,* Villeneuve d'Ascq, France, 6 pp., July 2010.
11. M. D. Doan, T. Keviczky, B. De Schutter: An Iterative Scheme for Distributed Model Predictive Control Using Fenchel's Duality, Journal of Process Control 21, 2011, pp. 746–755.

12. H. Scheu and W. Marquardt: Sensitivity-Based Coordination in Distributed Model Predictive Control, Journal of Process Control, 2011, doi: 10.1016/j.jprocont.2011.01.013
13. D. Jia and B. H. Krogh: Distributed Model Predictive Control, Proceedings of the American Control Conference Arlington, VA June 25-27, 2001, pp. 2767-2772.
14. P. D. Christofides, J. Liu, D.M. de la Pena: Networked and Distributed Predictive Control, Methods and Nonlinear Process Network Applications, Springer-Verlag London Limited, 2011

List of used symbols and abbreviations

Abbreviations

MPC: model predictive control

DMPC – SC: distributed model predictive control – stability constrained

LSS: large scale systems

LS: large scale

IBM: Interaction Balance Method

DMPC: distributed model predictive control

KKT: Karush–Kuhn–Tucker

DH: distributed Han's algorithm

IDH: improved distributed Han's algorithm

MIDH: modificated improved distributed Han's algorithm

Section 2

$\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}$:state, control, and output vectors

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$:system, control\input, output matrices

$\boldsymbol{Q}, \boldsymbol{R}$:positive definite diagonal weighting matrices

k: discrete time

$\boldsymbol{x}_c$: calculated state vector

$\boldsymbol{x}_p$: predicted state vector

$\boldsymbol{y}_p$: plant output vector

$\boldsymbol{y}_M$: model output vector

$\boldsymbol{E}$: error between plant and model output vector

$\boldsymbol{R}$: reference vector

Section 3

N: number of subsystems

$\hat{\boldsymbol{x}}_i$: conditioned optimal state vector

$\hat{\boldsymbol{v}}_i$: conditioned optimal control vector

$\boldsymbol{x}_i^*$: optimal state vector

$\boldsymbol{v}_i^*$: optimal control vector

$\boldsymbol{\lambda}$: Lagrange multiplier

$\boldsymbol{\lambda}^*$: optimal Lagrange multiplier

$L_i(\cdot)$: Lagrange function

$\hat{L}_i(\cdot)$: conditioned optimal Lagrange function

J: cost function

J^*: optimal cost function

$\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{C}_i$: state, control, and coupling matrices for subsystem i

$\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{v}_i$: state, control, and pseudo-control vectors

$\boldsymbol{K}$: coupling matrix

$\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{S}$:positivedefinite diagonal weighting matrices

$\overline{\boldsymbol{Q}}$: definite diagonal weighting matrix only for end state

t_f: end time

$\boldsymbol{e}^k$: total error vector

α^k: step size

$\Phi(\boldsymbol{\lambda})$: dual function

t: time

Section 4

$\boldsymbol{N}$: gradient matrix

$\boldsymbol{g}_a$**:** vector of active constraint

$\boldsymbol{S}$**:** search direction

$\boldsymbol{P}$**:** projection matrix

$\boldsymbol{\lambda}$**:** Lagrange multiplier

q: convex function

C: closed convex set

$\boldsymbol{H}$: Hessian matrix

s: number of the constrains

n_{eq} : number of equality constraints

n_{ineq} : number of inequality constraints

$q^*(\boldsymbol{y})$: conjugate function

$x^{(p)}$:values of the optimization vector at the p^{th} iteration

M:number of local controllers

$\boldsymbol{\gamma}_l$:intermediate variable

k_α: scaling vector

$\bar{h}_i$: average weight

K: number of intervals

L_i:the set of indices o constraints that subsystem i is responsible for updating their dual variables throughout the algorithm

$\mathcal{N}^i$:the neighborhood of subsystem i, consisting of i itself and other subsystems that have direct dynamical or constraints couplings with subsystem i

$L_{\mathcal{N}^i}$:the set of indices of constraints within responsibility of all subsystems in $\mathcal{N}^i$

$\boldsymbol{x}^{(p)|i}$:the self image of the global variable vector $\boldsymbol{x}^{(p)}$ made by subsystem i

$\boldsymbol{x}^{(p)|\mathcal{N}^i}$:the neighborhood image of $\boldsymbol{x}^{(p)}$ made by subsystem i

$\boldsymbol{x}^{(p)|\mathcal{N}^i}_{assumed}$:the assumed neighborhood image made by subsystem i

J^i:index matrix of subsystem i

Section 5

$\boldsymbol{x}_i^j, \boldsymbol{u}_i^j, \boldsymbol{v}_i^j, \boldsymbol{z}_i$: state, control, pseudo-control, and consumption vectors

k: number of discrete intervals

a_{12}, a_{23}: connection coefficient

$\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{S}$:positive definite diagonal weighting matrices

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{Z}$: state, control, coupling, consumption vectors

$\tilde{\boldsymbol{x}}_i$: difference between actual state and its setpoint

$\boldsymbol{x}_{s,i}, \boldsymbol{u}_{s,i}, \boldsymbol{v}_{s,i}$: setpoints for state, control, and pseudo-control

Author's statement

Hereby I declare that the present thesis was prepared by me and none of its contents was obtained by means that are against the law. The thesis has never before been a subject of any procedure of obtaining an academic degree. Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

_____01.10.2012_____ ____Lazutkin____

Date Author's signature

(B.Sc. Evgeny Lazutkin)